INTEGRATION OF MACHINING INSPECTION SENSORS AND SOFTWARE

INTEGRATION OF MACHINING INSPECTION SENSORS AND SOFTWARE

By

D. ALAN SAWULA, B. ENG

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfilment of the Requirements
for the Degree
Master of Applied Science

McMaster University

MASTER OF APPLIED SCIENCE (2013)

McMaster University

(Mechanical Engineering)

Hamilton, Ontario

TITLE: Integration of Machining Inspection Sensors and Software.

AUTHOR: D. Alan Sawula, B. Eng (McMaster University)

SUPERVISOR: A. D. Spence

NUMBER OF PAGES: xii, 108

Abstract

Ideally, the nominal design of a part or assembly, created with 3D Computer-Aided Design and Manufacturing (CAD/CAM) software, can be consistently fixtured and machined. In reality, process conditions vary, and feedback and correction methods such as integrated on-machine inspection, analysis, and process adjustment, are required.

On-machine inspection based on touch trigger probes is well established, but limited motion control computing capability restricts analysis to simple arithmetic. This prevents on-line use of known whole part mathematical analysis software that implements the part salvaging intentions of modern Geometric Dimensioning and Tolerancing (GD&T) standards. Additionally, no CNC integrated method exists for geometrically adjusting nominal tool paths so that an in-tolerance final part is produced. Machine tool support for high data rate sensors such as laser scanners is also lacking.

This thesis reports progress towards bidirectional integration of machine tool mounted inspection sensors with GD&T analysis software, and subsequent toolpath adjustment. The concepts are demonstrated using a fixture consisting of three datum spheres and a workpiece. The fixture is clamped in the CNC machine, datum spheres are measured, and after mathematical data fitting and registration, an in-tolerance final part is

produced. To facilitate multiple tests, a tool path is split into four and machined in four poses with measurement and tool path adjustment for each pose. Preliminary integration of a laser scanner with axis scales and computer software was also accomplished.

Acknowledgements

I would like to thank my supervisor Dr. Spence, for his patience, generosity, and support over our years working together. I have had the opportunity to meet and work with many interesting and intelligent people, and I thank Dr. Spence for giving me that opportunity. When faced with setbacks, Dr. Spence made sure we found a solution, and that both textbook and life lessons were learned. I am grateful for his dedication.

None of this work would be possible without the following people. Terry Wagg, for CNC training, Joe Verhaeghe for electronics assistance, and Jim McLaren, Mark MacKenzie and Ron Lodewyks for help in the machine shop.

Thank you to Yu Pin Lin and Hazem Mazhar for their assistance with laser scanning equipment and software.

Funding support from NSERC Canadian Network for Research and Innovation in Machining Technology (CANRIMT) is sincerely appreciated.

To my family and friends for all their unconditional support I am very grateful.

To Lisa, your patience and encouragement, are more than I could ever ask for.

Table of Contents

Chapter 1 Introduction 1

Chapter 2 Literature Review 6

2.1 Measurement in the Manufacturing Process 6

2.1.1 Static Sensor Specifications 9

2.1.2 Touch Trigger Probes 11

2.1.2.1 Kinematics of Touch Trigger Probes 13

2.1.2.2 Relevant CNC G-Codes for Touch Trigger Probes 15

2.1.2.3 G01 Linear Motion Command 15

2.1.2.4 G31 Skip Command 16

2.1.3 Laser Digitizers 16

2.2 Data Fitting 21

2.2.1 Rigid Coordinate System Registration 21

2.2.2 Orthogonal Least Squares Geometric Fitting 23

2.2.3 Orthogonal Least Squares Fitting on Data Sets 26

2.2.4 Commercial Point Cloud Software 28

2.3 Summary 29

Chapter 3 Hardware Description 30

3.1 CNC Machine Tool 30

3.2 Touch Trigger Probe Integration 30

3.3 Hardware Overview 32

3.4 Ax9150 UMI 33

3.4.1 Installation 34

3.4.2 Passing coordinate data to PC 34

3.5 Calibration of Touch Trigger Probe 35

3.5.1 Calibration Tool Path 36

3.5.2 Results of Calibration Data 38

3.6 Three Sphere Probing Routine 39

Chapter 4 Registration Experiment 42

4.1 Experiment Tool Path 42

4.2 Tool Path Adjustment 43

4.2.1 Part Registration in Manufacturing Processes 43

4.2.2 Tooling Sphere Registration Method 43

4.3 Test Cases 45

Chapter 5 Registration Experiment Results 49

5.1 Laser Scan of Finished Workpiece 49

5.2 Analysis of Expected Errors 59

Chapter 6 Laser Scanner Integration 61

6.1 LMI Scan Head Integration 63

6.2 Mount Design 64

6.3 Desktop Control Software 66

6.4 Expected Errors 71

6.5 Test Data 72

Chapter 7 Conclusions and Future Work 78

7.1 Future Work 79

References 109

Figures and Tables.

Figure 2.1 - Accuracy, Repeatability and Resolution in 2 example sensors, possible accuracy improvement shown for Sensor 1 10

Figure 2.2 - Touch Probes [23] : (a) TPS1(S) Touch Trigger Probe; (b) Touch Trigger Probe Principle of Operation; (c) SP600 Analog Touch Probe 12

Figure 2.3 – Touch Trigger Probe Kinematics - Low Force Direction [24] . 14

Figure 2.4 – Touch Trigger Probe Kinematics - High Force Direction [24] 14

Figure 2.5 - Pre-Travel Variation in Resistive Touch Trigger Probes 15

Figure 2.6 - Triangulation using Two Cameras [26], (b) Structured Light [27] 17

Figure 2.7 - Specular and Diffuse Reflection in laser scanning systems 18

Figure 2.8 - Limitations of Laser Scanners 19

Figure 2.9 - Laser Line with Camera Digitizer [30] 20

Figure 2.10 - 3 Point Registration MCS→CAD 22

Figure 2.11 - Orthogonal Least Squares Planar Circle Parameters 23

Figure 3.1 - Proposed Factory Floor Architecture 32

Figure 3.2 - Memex Ax9150 Universal Machine Interface [38] 34

Figure 3.3 - Ring Gauge Probing Path. Arrows indicate 8 of 36 trajectories. 37

Figure 3.4 - 2 Hit Probing Method 38

Figure 3.5 - Graph of Pre-travel variation using 36 trajectories 39

Figure 3.6 - 5 Point Probing of a Sphere 40

Figure 4.1 - Surface Finishing Operation Toolpath 42

Figure 4.2 - Fixture Base with Tooling Spheres and Workpiece 44

Figure 4.3 - Toolpath Case Divisions 45

Figure 4.4 - Machining Case 0 47

Figure 5.1 - Inspection Report of Experiment Case 0, Roland Scan Data .. 50

Figure 5.2 - Case 0, Roland Scan Data – Colour Map adjusted to show detail. 51

Figure 5.3 - Inspection Report of Experiment Cases 1-4, Roland Scan Data 52

Figure 5.4 – Cases 1-4, Roland Scan Data – Colour Map adjusted to show detail. 53

Figure 5.5 - Photograph of Machined Experiment Workpiece 58

Figure 6.1 - Gocator 2330 Specifications Diagram [44] 62

Figure 6.2 - CMM-Laser Scanner Hardware Architecture Diagram 64

Figure 6.3 - 3D Scanner Mount with bolt hole pattern for locking pin 65

Figure 6.4 – LMI 3D Gocator 2300 Scanner on an IOTA P CMM 66

Figure 6.5 – Flow Chart of PC Application Scanning Function 68

Figure 6.6 - Geometric Fit of Captured Point Cloud with 0.0684 mm error in diameter. 69
Figure 6.7 – CMM Laser Scanner Application Graphical User Interface .. 70
Figure 6.8 - LMI Scan Data: Case 0 Surface Deviation Analysis 73
Figure 6.9 - Case 0, LMI Scan Data – Colour Map adjusted to show detail. 74
Figure 6.10 - LMI Scan Data: Cases 1-4 Surface Deviation Analysis 75
Figure 6.11 – Cases 1-4, LMI Scan Data – Colour Map adjusted to show detail. 76

Table 4.1 - OLS Sphere Fit 47
Table 4.2 - Fitted Sphere Centre Registration Error and Registration Components 48
Table 5.1 - Experiment Inspection Data (Roland) Statistics 54
Table 5.2 - Sphere Fit Statistics 55
Table 5.3 - Sphere Centre Distances 56
Table 5.4 - Plane Fitting Data 57
Table 5.5 - Summary of Expected Errors 60
Table 6.1 - Gocator 2330 Laser Scanner Specifications 62
Table 6.2 - CMM Laser Cumulative Error 72
Table 6.3 - LMI Scan Data Surface Fit Statistics 77

List of Abbreviations and Symbols

NC – Numerical Control
CNC – Computer Numerical Control
3D – Three Dimensional
CAD – Computer Aided Design
CAM – Computer Aided Manufacturing
CMM – Coordinate Measuring Machine
GD&T – Geometric Dimensioning and Tolerancing
HTM – Homogeneous Transformation Matrix
MCS – Machine Coordinate System
OLS – Orthogonal Least Squares
SVD – Singular Value Decomposition
ICP – Iterative Closest Point
CME – Checkmate Engine
DLL – Dynamic Link Library
PCL – Point Cloud Library
UMI – Universal Machine Interface
FOV – Field of View

Chapter 1

Introduction

Numerical control (NC) machining is a method of automatically operating a manufacturing machine based on instruction codes. NC machining offers advantages over manual production such as better control of cutting parameters, improved part quality and repeatability, reduced manufacturing time and reduced scrap [1].

Computer numerical control (CNC) brings the local storage and connectivity capabilities of a computer system to NC machining, replacing the punched tape and dedicated hardware of early NC systems.

The essence of conventional computer-aided manufacturing automation is repeatability. In CNC machining workflow, a part design created using three dimensional (3D) Computer-Aided Design (CAD) software can be interpreted by Computer-Aided Manufacturing (CAM) software to generate cutting tool paths. The part is then clamped into place, the tool paths executed, and the desired final shape is produced.

In practice, many factors add variability to the machining process, such as:

- In Process Variability

The cutting tool may be worn or break unexpectedly. If this can be detected before the part is damaged, a replacement tool can be used to continue the operation [2].

- Geometric/Setup Variability

  Examples include casting variations, stamping die / injection mould heat treat distortion, and fixturing setup variability. For refurbishment of turbine engine compressor blades, in service creep and weld bead shape differs in each case. Measurement and nominal tool path adjustment for each blade is therefore required [3].

Presently, the steps of measurement, analysis, toolpath adjustment, and machining are accomplished with separate tools lacking direct integration and automation.

Part measurement in manufacturing is commonly performed as, but not limited to, a post-process inspection. Parts are measured on dedicated gaging systems or Coordinate Measuring Machines (CMM) to verify dimensions against nominal geometric dimensioning and tolerancing (GD&T) specifications. This thesis focuses on processes using CMM systems to perform part measurement. CNC and CMM design, calibration and error compensation have long been studied, changing by a small amount in comparison to the advancements in machine control computational ability, software, and the measurement devices used on these machines [4]. Resistive

technology based touch trigger probes, while popular in use, have been surpassed by piezoelectric and scanning touch trigger probes that offer significant improvements. Non-contact measurement sensors have entered as a fast and accurate alternative to touch based measurement for some applications.

Inspection data analysis software packages are relied on for verification of part dimensions. The output of the inspection, such as a surface deviation report, is used for approval of parts, and also as feedback for the manufacturing process. Unfortunately, the inspection process is usually performed after machining, and thus after any machining errors have occurred. Recent analysis software such as OriginSoftFit [5] offers a proactive approach with the ability to inspect a part prior to machining and make modifications to the part program based on inspection data. Such modifications may include global rigid transformations, or even per-feature toolpath adjustments such as removing refurbishment weld material [6]. After use of this software, typically employed with a dedicated CMM or laser digitizer, the part and toolpaths must be transferred a machine tool to produce the final part.

Common machine tool sensors include tool setters and touch trigger probes. Tool setters are used to determine the tool length, and touch trigger probes are used to locate the workpiece within the machine workspace.

Virtually all turning centres and machining centres come equipped with some form of touch trigger probe option [7]. Although the CNC control computer is quite capable of performing the mathematics required for interpreting numerical control (NC) code, it cannot run least squares or other data fitting techniques with the same efficiency as a modern computer used for processing CMM inspection data.

The lack of integrated and automated methods for using these tools creates opportunity for operator error in recording, entering and analysis of data [4], discouraging adoption of the tools. Because of this, scrap production may result from a failure to adjust nominal tool paths to actual part geometry, even though, by using available GD&T software, a salvaging opportunity exists. This is the primary motivation of the thesis.

The objective of this thesis is to describe and develop methods allowing analysis and tool path adjustment software to be integrated with machine tool based measurement, so that solutions to the problem of geometric variability in machining can be provided. Additionally, the thesis aims to integrate a laser scanner with exisiting CMM equipment to investigate the feasibility of integrating a laser scanner on a CNC machining centre.

A literature review is presented in Chapter 2, and the hardware architecture is discussed in Chapter 3. Chapter 4 presents an experiment where additional CNC hardware, and geometric analysis of measured datums

were used to update nominal tool paths to match the measured workpiece location, and an in-tolerance part was produced. Experimental results are discussed in Chapter 5. Integration of a laser scan head with a CMM is presented in Chapter 6, and conclusions and future work are discussed in Chapter 7.

Chapter 2

Literature Review

2.1 Measurement in the Manufacturing Process

There are many tools available for measurement in manufacturing processes, the choice of which to employ depends on many factors including production quantity, tolerance level, part complexity, and part variety. When dealing with high production quantities and low part variety, dedicated automatic gauges can be faster and more cost effective than CMM machines. When flexibility is desirable, such as with low to medium volumes and a variety of part geometries, or initial gauge qualification, the investment in a CMM can be justified [8]. This thesis focuses on inspecting and machining parts in volumes as low as a single part.

A scheme has been developed for classifying measurements based on their existence in the manufacturing timeline [9]. The four categories are preprocess measurement, in-process measurement, process-intermittent measurement and postprocess measurement. The remainder of this section discusses the involvement of automated inspection methods in each of the four categories.

Preprocess measurements are made on the manufacturing system itself. Calibration and correction of machine tools and measuring machines

fall in this category [4]. The ISO 10360 standard [10] describes "Acceptance and reverification tests for coordinate measuring machines", likewise the ISO 230-2 standard [11] evaluates the accuracy and repeatability of a machine tool, and the ISO 230-10 standard [12] evaluates the measuring performance of probing systems of numerically controlled machine tools. These standards test the machine for environmental sensitivity, displacement accuracy, spindle errors, thermal errors, kinematics, and cutting performance where applicable [4]. When properly characterized, controlled and corrected, machine tools can approach the accuracy of a measuring machine [13], [14], [15] as cited in [4].

In-process measurements are measurements taken while the feature is being produced. These measurements may be directly measured, or indirectly measured through "deterministic metrology", where process variables other than the attribute of interest (dimension) are monitored in order to give information about the attribute of interest. Direct measurement faces many impending hazards on the factory floor such as chips, coolant and the tool itself [16]. The most successful indirect in-process measurements pre-calibrate the machine and monitor on-line variables, correcting the process according to machining error-process variable relationship models [17]. Laboratory implementations have demonstrated successful monitoring of

machining errors by monitoring machine temperature, cutting forces, and vibrations, however this technique has yet to reach the factory floor [18].

Process-Intermittent measurements are performed during a momentary break from machining where measurements are taken and used to modify process parameters. This may be performed on or off the machine using a touch-trigger probe, or by other gauging equipment such as hand-held device. Higher end CMM systems employ analog scanning probes capable of tracing surface contours, but point measurement based touch trigger probes are much more common among machining centres. The use of a touch probe on a machining centre has many sources for error that complicate measuring. The steady state temperature conditions that are imposed on CMM machines cannot be imposed on CNC machining centres. As a result, if the temperature of critical structural components is changes between machining and probing, the spindle position from machining to probing can be very large [19]. Additionally, there is the systematic aspect of measuring a part on the same equipment that machined it. If a systematic error exists with the machine at the time of machining, the same error will exist during inspection [20]. If these sources of error are understood and diligently considered, and machine tool measurement evaluation standards are followed, is has been shown that on-machine probing has a valid place in the manufacturing process.

Postprocess measurements are the measurements performed after machining. Depending on the process these measurements could result in pass/fail decisions, further machining, and/or correction of process parameters.

2.1.1 Static Sensor Specifications

Manufacturing measurement sensors and systems are typically described based on their static performance specifications since part dimensions are expected to be constant over the duration of measurement. Accuracy, repeatability and resolution are common specifications used to describe sensor capabilities. Repeatability, also referred to as precision, is the:

'closeness of agreement between indications or measured quantity values obtained by replicate measurements on the same or similar objects under specified conditions' [21].

The standard deviation range that the repeatability value represents must also be specified. A 1 σ repeatability is achieved 68.27% of the time, while 2 σ and 3 σ are achieved 95.45% and 99.73% of the time. By the same source, Accuracy is defined as the:

'closeness of agreement between a measured quantity value and a true quantity value of a measurand'.

A high degree of accuracy in multiple sensor measurements cannot be achieved without a high degree of repeatability. Calibration can improve accuracy in sensors with a high degree of repeatability, but a sensor with a low degree of repeatability will have inherently lower accuracy even after calibration. Resolution is defined as the:

'smallest change in a quantity being measured that causes a perceptible change in the corresponding indication'.

Resolution is important, but a higher resolution cannot compensate for deficiencies in accuracy. The concepts of accuracy, repeatability and resolution are illustrated in Figure 2.1.

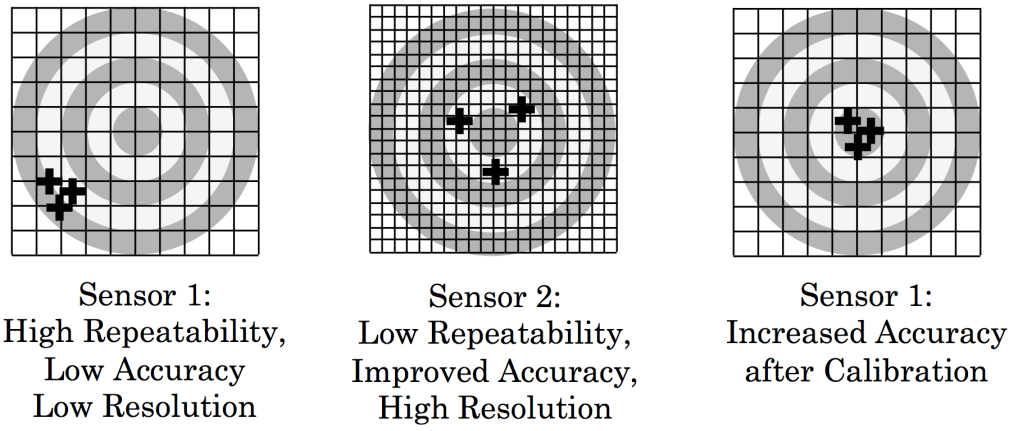
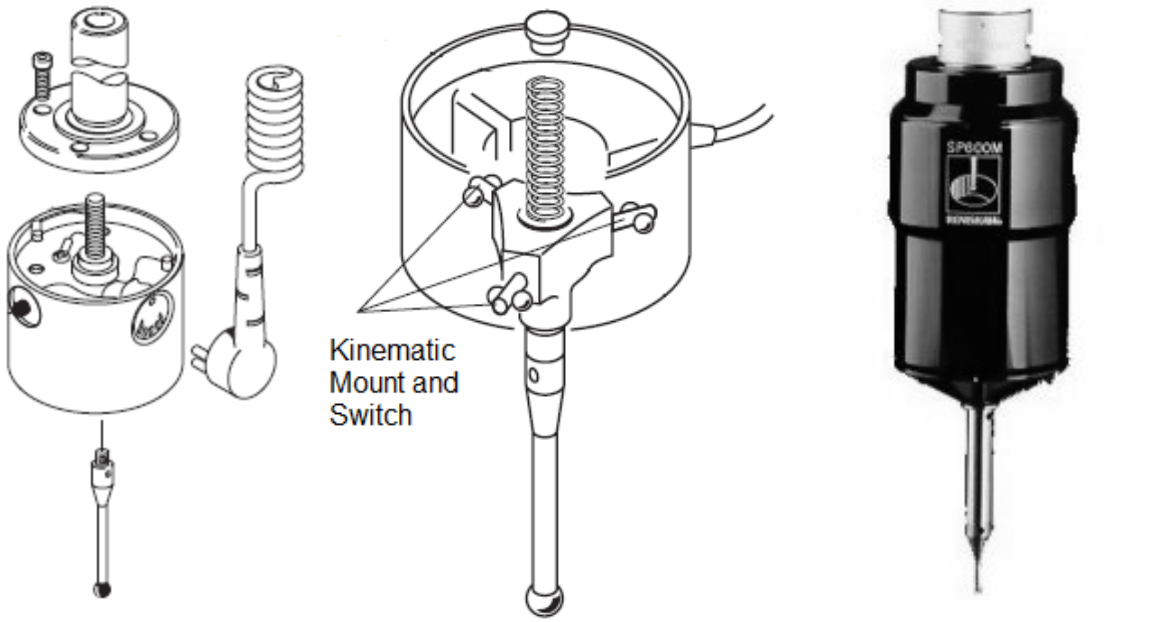


Figure 2.1 - Accuracy, Repeatability and Resolution in 2 example sensors, possible accuracy improvement shown for Sensor 1

2.1.2 Touch Trigger Probes

The touch trigger probe (Figure 2.2(a)) was invented to measure fuel lines on the Concorde supersonic aircraft, where existing hard contact methods caused the fuel lines to deflect during measurement [22]. The touch trigger probe relies on three equally spaced contact points, serving as series wired normally closed switches (Figure 2.2(b)). The probe is positioned in the X, Y, and Z directions using CMM or CNC machine tool motorized axes, and upon making contact with the part being measured, one of three switch contacts will open. The open switch signals the control computer to stop motion, and save the current axes positions in memory. The positions can then be used for mathematical calculations. Before another point can be measured, the probe must be moved away from the point of contact to allow the probe switch to close, and to avoid damage to the probe due to excess deflection. Dynamic effects limit the overall data collection rate to approximately one point per second. Newer analog touch probes (Figure 2.2(c)), available for CMMs, measure the amount of stylus deflection and can obtain continuous measurements by maintaining contact with the surface.



(a) **(b)** **(c)**

Figure 2.2 - Touch Probes [23] : (a) TPS1(S) Touch Trigger Probe; (b) Touch Trigger Probe Principle of Operation; (c) SP600 Analog Touch Probe

Touch trigger probes are also available for CNC machine tools. The probe can be loaded into the spindle using an automatic tool changer, and replaced with a cutting tool for machining. A wireless communication link (infrared or radio frequency) is used to transmit the probe switch open signal. As with CMM touch trigger probes, the data collection rate is slow. The machine tool control has limited mathematical calculation ability, and lack of direct integration prohibits use of analog probes with standard CNC controls.

2.1.2.1 Kinematics of Touch Trigger Probes

The proper use of a touch probe requires the understanding of pre-travel and probe lobing concepts. Pre-travel refers to the amount the stylus deflects as it generates the torque required to unseat the probe. The stylus deflection **d** is dependent on applied force **F** according to the bending stiffness of a cantilever beam modelled with cross-sectional moment of inertia **I**, elastic modulus **E,** and length **P.**

$$d = \frac{P^3 F}{3EI} \tag{2.1}$$

Since the stylus length **P** dominates the deflection, one should use the shortest stylus possible while maintaining the ability to visit all the required probe locations, such as those deep inside a hole.

The concept of probe lobing refers to the direction dependent pre-travel variation shown in Figure 2.3 and Figure 2.4. To unseat the probe, the stylus torque $\boldsymbol{T_s}$ created by the force $\boldsymbol{F_s}$ at the stylus tip must overcome the torque $\boldsymbol{T_k}$ created by the spring force $\boldsymbol{F_k}$ and distance $\boldsymbol{R}$. The torque will be lowest when the pivot point is the line drawn between two of the stylus supports, as the perpendicular distance to the stylus axis is minimal. This effect will contribute an error as if the tip were not spherical, as shown by the dotted line in Figure 2.5. Since the effects of pre-travel are repeatable, calibration can be applied to increase the accuracy of touch trigger probes.

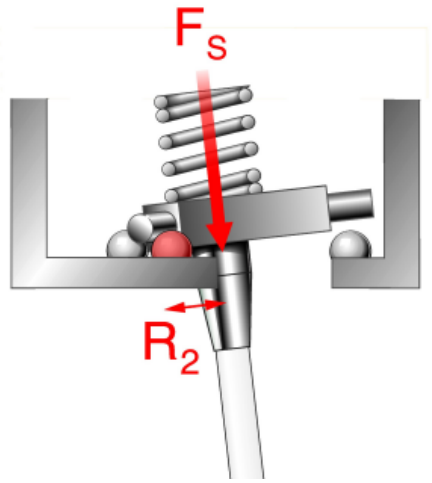


Figure 2.3 – Touch Trigger Probe Kinematics - Low Force Direction

[24]

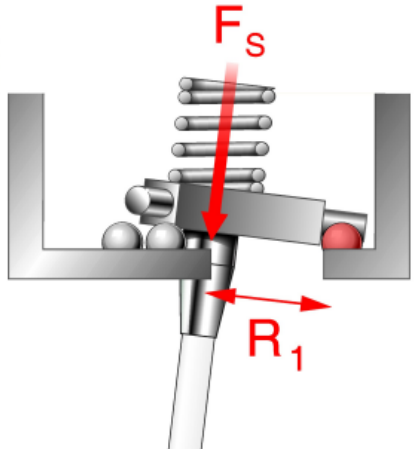


Figure 2.4 – Touch Trigger Probe Kinematics - High Force Direction

[24]

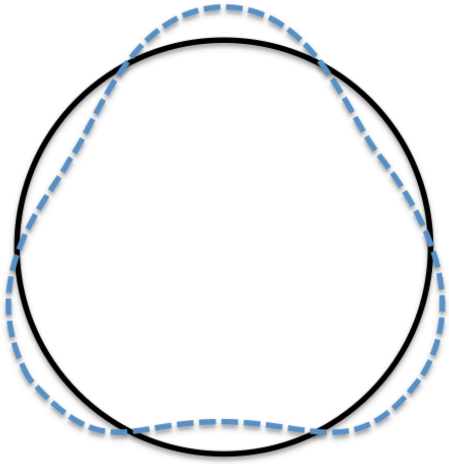


Figure 2.5 - Pre-Travel Variation in Resistive Touch Trigger Probes

2.1.2.2 Relevant CNC G-Codes for Touch Trigger Probes

Operation of a touch trigger probe on a CNC machine tool requires a sequence of programmed motion commands. The Fanuc Series 15i-MA control used in this thesis used the RS274D G-code standard. The essential G-code commands and syntax are given in sections 2.1.2.3 and 2.1.2.4 .

2.1.2.3 G01 Linear Motion Command

The **G01X**xx.xxx**Y**yy.yyy**Z**zz.zzz**F**ff.fff linear interpolation command is used to move along a straight line from the current position to (xx.xxx,yy.yyy,zz.zzz) at feed rate ff.fff. If the G21 code specifying programming in mm is active, positions are specified in mm, and feedrates in

mm/min. The G01 code is used to position the probe before measurement, and to back away from the part surface after measurement.

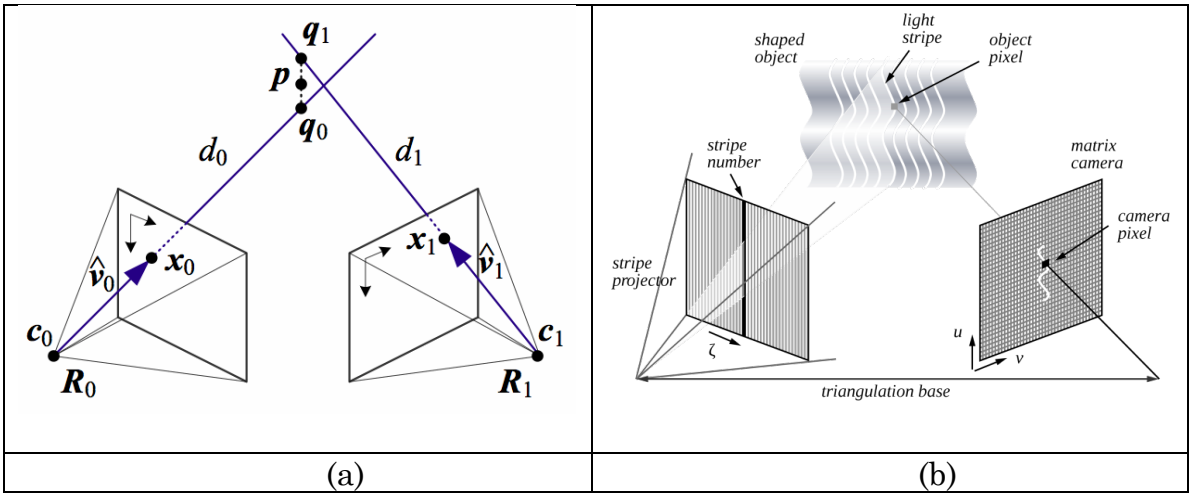
2.1.2.4 G31 Skip Command

The touch trigger probe is electrically connected to the CNC motion control so that a special skip signal is generated when the probe switch opens upon contacting the workpiece. To interface with the skip signal, the **G31X**xx.xxx**Y**yy.yyy**Z**zz.zzz**Fff.**fff command is used. This command is similar to **G01**, in that linear motion begins at the current position and proceeds to (xx.xxx,yy.yyy,zz.zzz) at feed rate ff.fff. If the skip signal is received, motion halts and the CNC control records the machine coordinate system (MCS) position to local variables. The Fanuc 15i-MA control used in this work assigns the X,Y and Z MCS positions to local variables #5061, #5062 and #5063 [25]. Once MCS positions are stored, the next line in the NC code will be executed. Since the skip signal remains high while the probe is in contact, must be backed away from the surface using a command other than G31.

2.1.3 Laser Digitizers

When data collection rates of thousands of points per second are needed, non-contact laser digitizers have emerged as the preferred solution. Triangulation is a technique used by 3D scanning systems (Figure 2.6 (a)) to

determine a 3D point location based on its location in two or more 2D images with known camera locations and orientations [26].



(a) (b)

Figure 2.6 - Triangulation using Two Cameras [26], (b) Structured Light [27]

The identification of pixels $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ in Figure 2.6(a) that correspond to the same real-world point $\boldsymbol{p}$ is known as the correspondence problem, and is a very difficult computer vision problem [28]. One strategy to assist in pixel identification is the use of structured light techniques (Figure 2.6(b)). Using a projector or laser diode, a line can be projected from viewpoint $\boldsymbol{R}_0$. The point $\boldsymbol{p}$, identified in $\boldsymbol{R}_1$ as pixel $\boldsymbol{x}_1$, must exist at the intersection of $\hat{\boldsymbol{v}}_1$ and the plane formed by the projected line.

Figure 2.9 shows a laser scanner employing these techniques, and the laser line as seen by the camera. Synchronization and calibration with the machine axes is required to obtain the locations in a fixed reference frame.

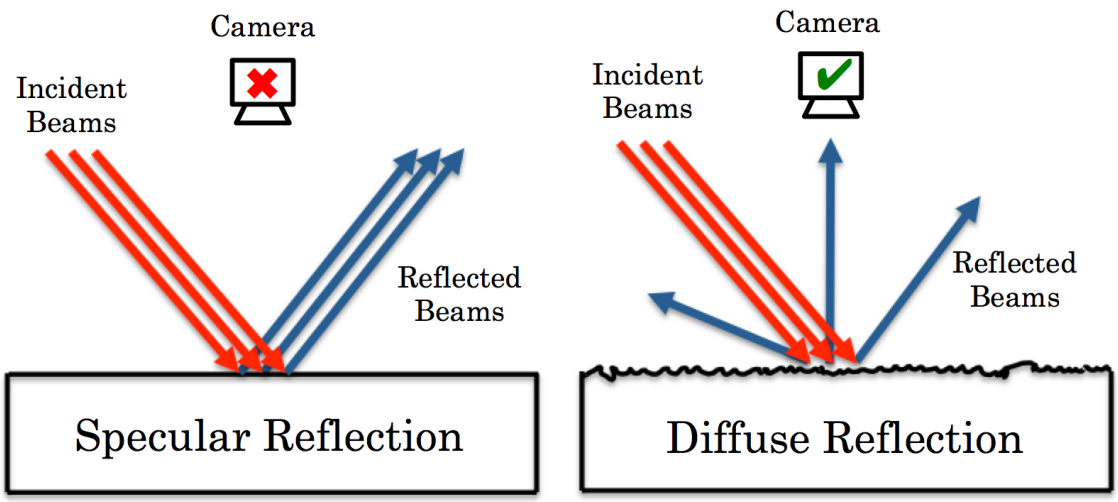


Figure 2.7 - Specular and Diffuse Reflection in laser scanning systems

Figure 2.3 illustrates the issue of specular reflection in laser scanning systems, where no light is reflected from the surface toward the camera [29].

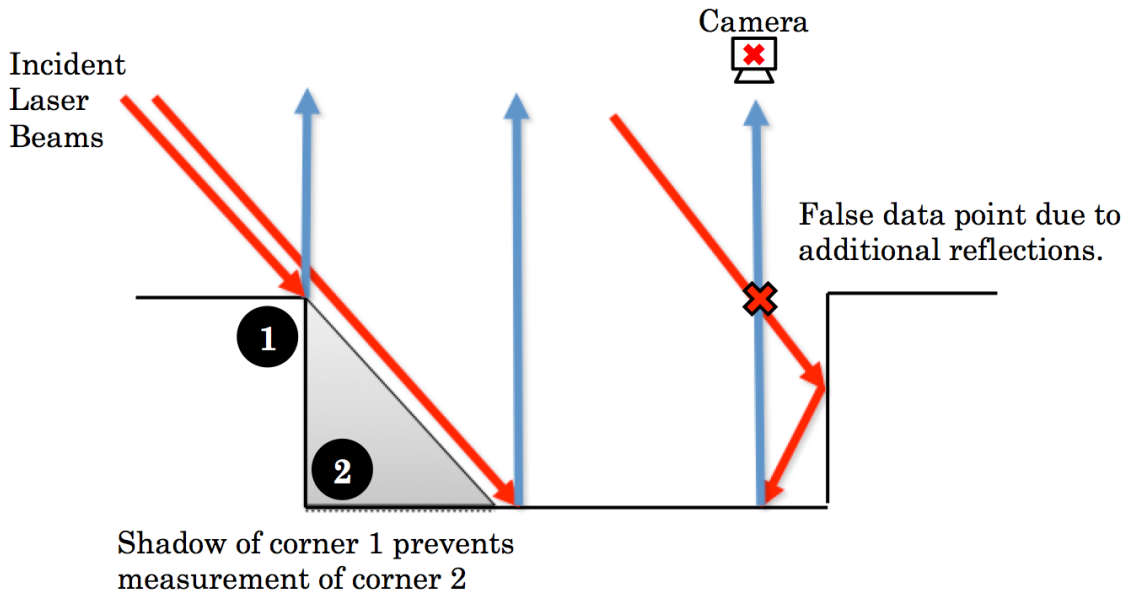


Figure 2.8 - Limitations of Laser Scanners

Figure 2.8 illustrates two limitations of laser scanners. On the left, the shadow caused by another part feature (corner 1) obstructs the path of the laser. To overcome this, multiple view angles are required, which can be obtained by reorienting the scanner and completing multiple scanning passes.

Also illustrated in Figure 2.8 is the false data scenario, where additional reflections are visible to the camera. Since some light is absorbed at each reflection, corrective actions include reducing the incoming beam strength to only allow once reflected beams to trigger the sensor. Additionally, if the light sensor stores the intensity value of the collected

point, intensity values that are significantly lower compared to points taken in the same area could be identified as false data.

Other limitations include high data rate transmission requirements, and the need for sophisticated mathematical processing. Nonetheless, this type of sensor is emerging as an important solution.

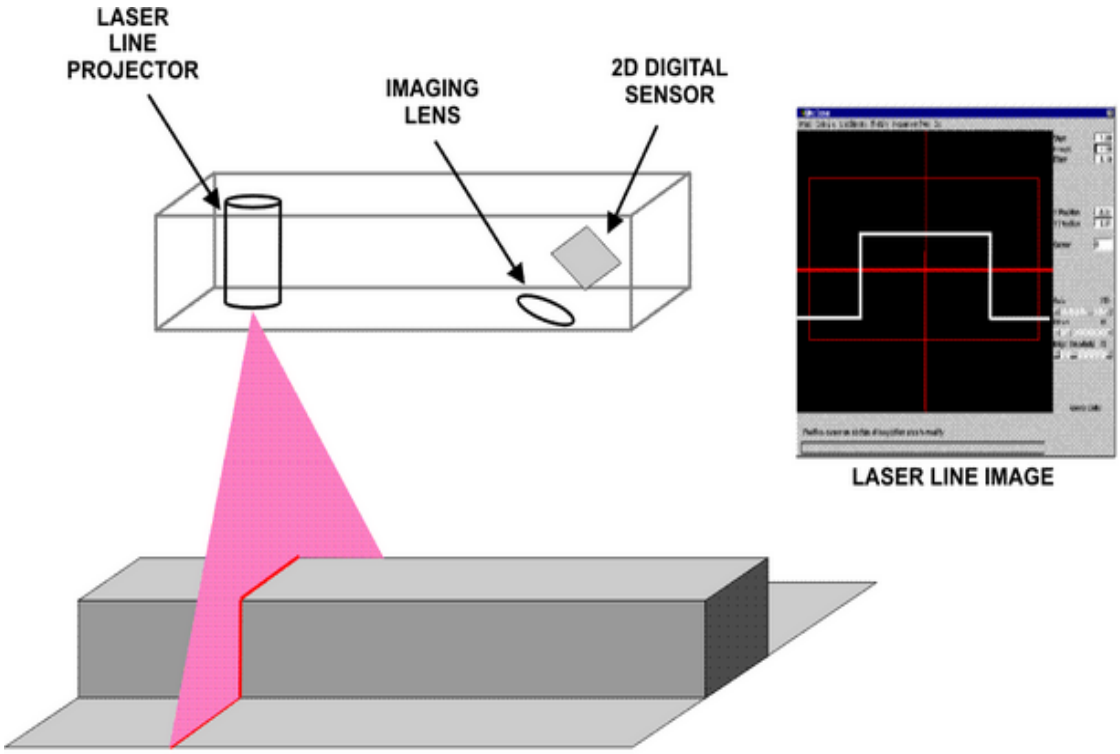


Figure 2.9 - Laser Line with Camera Digitizer [30]

2.2 Data Fitting

2.2.1 Rigid Coordinate System Registration

With any inspection sensor / machine, there is a need to align various coordinate systems. Data collected on a machine will be expressed using different coordinates than were used in the CAD representation. Conversion can be accomplished using Homogeneous Transformation Matrices (HTMs). A simple procedure is to measure $N = 3$ points (for example sphere centres) on the machine, and then map them to the corresponding CAD points. Denoting the machine coordinate system with subscript MCS, and the CAD coordinates with subscript CAD, the points are expressed as $\mathbf{P}_{i,\mathrm{MCS}}\left(x_{i,\mathrm{MCS}}, y_{i,\mathrm{MCS}}, z_{i,\mathrm{MCS}}\right)$, and $\mathbf{P}_{1,\mathrm{CAD}}\left(x_{1,\mathrm{CAD}}, y_{1,\mathrm{CAD}}, z_{1,\mathrm{CAD}}\right) = (0,0,0)$, $\mathbf{P}_{2,\mathrm{CAD}}\left(x_{2,\mathrm{CAD}}, y_{2,\mathrm{CAD}}, z_{2,\mathrm{CAD}}\right) = (1,0,0)$, $\mathbf{P}_{3,\mathrm{CAD}}\left(x_{3,\mathrm{CAD}}, y_{3,\mathrm{CAD}}, z_{3,\mathrm{CAD}}\right) = (0,1,0)$.

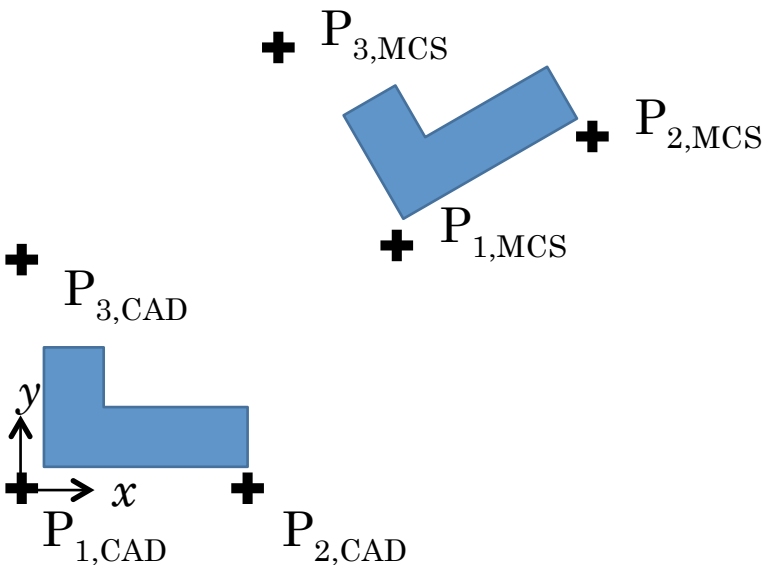


Figure 2.10 - 3 Point Registration MCS→CAD

Note that the misalignments have been greatly exaggerated for clarity.

It is assumed that a rigid transformation can be used with unit lengths and right angles. That is $\left\|\mathbf{P}_{2,\mathrm{MCS}}-\mathbf{P}_{1,\mathrm{MCS}}\right\|=\left\|\mathbf{P}_{2,\mathrm{CAD}}-\mathbf{P}_{1,\mathrm{CAD}}\right\|=1$, $\left\|\mathbf{P}_{3,\mathrm{MCS}}-\mathbf{P}_{1,\mathrm{MCS}}\right\|=\left\|\mathbf{P}_{3,\mathrm{CAD}}-\mathbf{P}_{1,\mathrm{CAD}}\right\|=1$, and $\angle\mathbf{P}_{3,\mathrm{MCS}}\mathbf{P}_{1,\mathrm{MCS}}\mathbf{P}_{2,\mathrm{MCS}}=\angle\mathbf{P}_{3,\mathrm{CAD}}\mathbf{P}_{1,\mathrm{CAD}}\mathbf{P}_{2,\mathrm{CAD}}=90^{\mathrm{o}}$.

With these assumptions, the HTM to convert from CAD to MCS can be written as

$$X=\left(P_{2,CAD}-P_{1,CAD}\right) \tag{2.2}$$

$$\boldsymbol{Y}=(\boldsymbol{P_{3,CAD}}-\boldsymbol{P_{2,CAD}}) \tag{2.3}$$

$$\boldsymbol{Z}=\left(\boldsymbol{P_{2,CAD}}-\boldsymbol{P_{1,CAD}}\right)\times(\boldsymbol{P_{3,CAD}}-\boldsymbol{P_{2,CAD}}) \tag{2.4}$$

$$\boldsymbol{HTM_{MCS\leftarrow CAD}}=\begin{bmatrix}\boldsymbol{X} & \boldsymbol{Y} & \boldsymbol{Z} & \boldsymbol{P_{1,CAD}}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1}\end{bmatrix} \tag{2.5}$$

The MCS to CAD HTM is the inverse, $\boldsymbol{HTM_{CAD\leftarrow MCS}}=\boldsymbol{HTM_{CAD\leftarrow MCS}^{-1}}$

2.2.2 Orthogonal Least Squares Geometric Fitting

After collecting the data points, such as from the surface of a sphere, a sphere must be mathematically fit to the data. Orthogonal Least Squares (OLS) methods were used. The fit parameters for an OLS circle fit are shown in Figure 2.11.

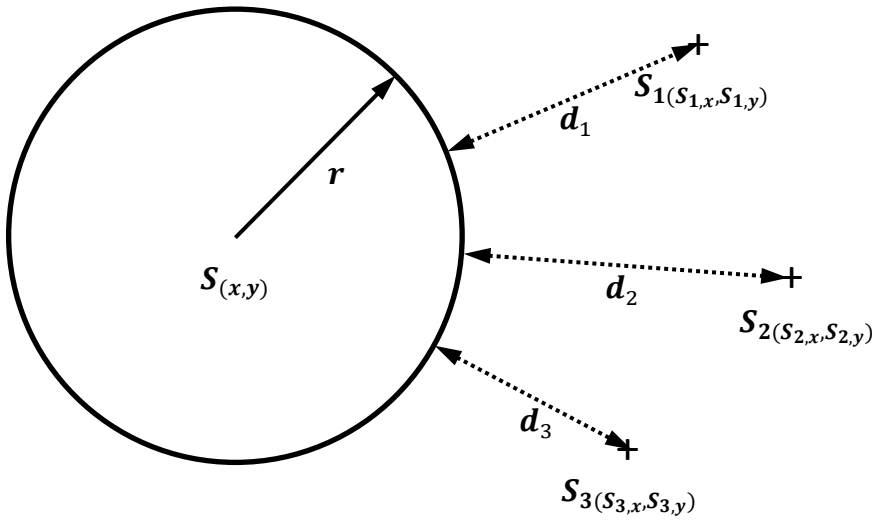


Figure 2.11 - Orthogonal Least Squares Planar Circle Parameters

For a sphere with sample points $\mathbf{S}_j$, centre point $\mathbf{S}$, and radius r, the orthogonal distance error is

$$d_j = \sqrt{\left(s_{j,x} - s_x\right)^2 + \left(s_{j,y} - s_y\right)^2 + \left(s_{j,z} - s_z\right)^2} - r \qquad \textbf{(2.6)}$$

and the OLS sum to minimize is $\Sigma^2 = \sum_{j=1}^{M} d_j^2$ where M is the number of sphere sample points. Assuming that the sphere is very nearly perfectly round (small form errors), a clever initial approximation is obtained using the linear approximation [31].

$$\begin{aligned} \theta_j &= \left[\left(s_{j,x} - s_x\right)^2 + \left(s_{j,y} - s_y\right)^2 + \left(s_{j,z} - s_z\right)^2\right] - r^2 \\ &= \left\{\left[\left(s_{j,x} - s_x\right)^2 + \left(s_{j,y} - s_y\right)^2 + \left(s_{j,z} - s_z\right)^2\right]^{1/2} - r\right\} \cdot \left\{\left[\left(s_{j,x} - s_x\right)^2 + \left(s_{j,y} - s_y\right)^2 + \left(s_{j,z} - s_z\right)^2\right]^{1/2} + r\right\} \\ &\approx d_j \cdot 2r \end{aligned}$$

and then minimizing the sum

$$\Theta^2 = \sum_{j=1}^{M} \theta_j^2 = \sum_{j=1}^{M} \left\{g_j - h - 2 \cdot \left[s_{j,x} s_x + s_{j,x} s_x + s_{j,x} s_x\right]\right\}^2$$

where $g_j = s_{j,x}^2 + s_{j,y}^2 + s_{j,z}^2$ and $h = r^2 - \left(s_x^2 + s_y^2 + s_z^2\right)$. Simultaneously solving the partial derivative equations

$$\frac{\partial \Theta^2}{\partial s_x} = 0 \quad \frac{\partial \Theta^2}{\partial s_y} = 0 \quad \frac{\partial \Theta^2}{\partial s_z} = 0 \quad \frac{\partial \Theta^2}{\partial h} = 0$$

leads to the linear system

$$\begin{bmatrix} \sum_{j=1}^{M} 1 & \sum_{j=1}^{M} s_{j,x} & \sum_{j=1}^{M} s_{j,y} & \sum_{j=1}^{M} s_{j,z} \\ \sum_{j=1}^{M} s_{j,x} & \sum_{j=1}^{M} s_{j,x}^2 & \sum_{j=1}^{M} s_{j,x}s_{j,y} & \sum_{j=1}^{M} s_{j,x}s_{j,z} \\ \sum_{j=1}^{M} s_{j,y} & \sum_{j=1}^{M} s_{j,x}s_{j,y} & \sum_{j=1}^{M} s_{j,y}^2 & \sum_{j=1}^{M} s_{j,y}s_{j,z} \\ \sum_{j=1}^{M} s_{j,z} & \sum_{j=1}^{M} s_{j,x}s_{j,z} & \sum_{j=1}^{M} s_{j,y}s_{j,z} & \sum_{j=1}^{M} s_{j,z}^2 \end{bmatrix} \cdot \begin{bmatrix} h \\ 2s_x \\ 2s_y \\ 2s_z \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{M} g_j \\ \sum_{j=1}^{M} s_{j,x}g_j \\ \sum_{j=1}^{M} s_{j,y}g_j \\ \sum_{j=1}^{M} s_{j,z}g_j \end{bmatrix}$$

that can be solved for h, s_x, s_y, and s_z. Newton-Raphson or equivalent iterative improvement is then used to minimize $\boldsymbol{\Sigma}^2$ in the exact problem by solving for δs_x, δs_y, and δs_z in the matrix equation

$$\begin{bmatrix} \frac{\partial \Sigma^2}{\partial s_x} \\ \frac{\partial \Sigma^2}{\partial s_y} \\ \frac{\partial \Sigma^2}{\partial s_z} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 \Sigma^2}{\partial s_x^2} & \frac{\partial^2 \Sigma^2}{\partial s_x s_y} & \frac{\partial^2 \Sigma^2}{\partial s_x s_z} \\ \frac{\partial^2 \Sigma^2}{\partial s_x s_y} & \frac{\partial^2 \Sigma^2}{\partial s_y^2} & \frac{\partial^2 \Sigma^2}{\partial s_y s_z} \\ \frac{\partial^2 \Sigma^2}{\partial s_x s_z} & \frac{\partial^2 \Sigma^2}{\partial s_y s_z} & \frac{\partial^2 \Sigma^2}{\partial s_z^2} \end{bmatrix} \cdot \begin{bmatrix} \delta s_x \\ \delta s_y \\ \delta s_z \end{bmatrix}$$

and then updating the sphere centre estimate using $s_x = s_x - \delta s_x$, $s_y = s_y - \delta s_y$, and $s_z = s_z - \delta s_z$. The improved radius estimate is $r = \frac{1}{M}\sum_{j=1}^{M} \sqrt{\left(s_{j,x} - s_x\right)^2 + \left(s_{j,y} - s_y\right)^2 + \left(s_{j,z} - s_z\right)^2}$. Iterations continue until the estimates for s_x, s_y, s_z, and r converge. Note that convergence is to a relative extreme, and this might be a maximum rather than a minimum.

Using the linearized initial approximation, with small form errors, reduces this likelihood. An example using experimental data is shown in Table 4.1.

2.2.3 Orthogonal Least Squares Fitting on Data Sets

If there are position repeatability errors $\boldsymbol{\eta}_i$ in the actual sphere locations, or there are more than four points, Orthogonal Least Squares (OLS) methods that distribute the error over the set are used [32]. For example, separating into rotation $\mathbf{R}_{\text{MCS}\leftarrow\text{PCS}}$ and translation $\mathbf{T}_{\text{MCS}\leftarrow\text{PCS}}$ components, the transformation with noise is

$$\mathbf{P}_{i,\text{MCS}} = \mathbf{R}_{\text{MCS}\leftarrow\text{PCS}}\mathbf{P}_{i,\text{CAD}} + \mathbf{T}_{\text{MCS}\leftarrow\text{PCS}} + \boldsymbol{\eta}_i \qquad \textbf{(2.7)}$$

The OLS goal is to find $\hat{\mathbf{R}}_{\text{MCS}\leftarrow\text{PCS}}$ and $\hat{\mathbf{T}}_{\text{MCS}\leftarrow\text{PCS}}$ that minimize the sum

$$\Sigma^2 = \sum_{i=1}^{N} \left\| \mathbf{P}_{i,\text{MCS}} - \left(\mathbf{R}_{\text{MCS}\leftarrow\text{PCS}}\mathbf{P}_{i,\text{CAD}} + \mathbf{T}_{\text{MCS}\leftarrow\text{PCS}} \right) \right\|^2 \qquad \textbf{(2.8)}$$

The solution is found using a Singular Value Decomposition (SVD) approach as follows. First calculate $\mathbf{P}_{\text{CAD}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{P}_{i,\text{CAD}}$ and $\mathbf{P}_{\text{MCS}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{P}_{i,\text{MCS}}$. Next define $\mathbf{Q}_{i,\text{CAD}} = \mathbf{P}_{i,\text{CAD}} - \mathbf{P}_{\text{CAD}}$ and $\mathbf{Q}_{i,\text{MCS}} = \mathbf{P}_{i,\text{MCS}} - \mathbf{P}_{\text{MCS}}$. Calculate $\mathbf{A} = \sum_{i=1}^{N}\mathbf{Q}_{i,\text{CAD}}\mathbf{Q}_{i,\text{MCS}}^{\text{T}}$, the SVD $\mathbf{A} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{V}^T$, and $\mathbf{X} = \mathbf{V}\mathbf{U}^T$. Usually the determinant $\det(\mathbf{X}) = +1$, and in this case $\hat{\mathbf{R}} = \mathbf{X}$, and $\hat{\mathbf{T}} = \mathbf{P}_{\text{MCS}} - \hat{\mathbf{R}}_{\text{MCS}\leftarrow\text{CAD}}\mathbf{P}_{\text{CAD}}$.

If the $\det(\mathbf{X}) = -1$, and an eigenvalue of $\mathbf{\Lambda}$ is zero, the reflection solution has been found. It can be corrected to a rotation by replacing $\mathbf{V}$ with $\mathbf{V}' = \begin{bmatrix} v_{1,1} & v_{1,2} & -v_{1,3} \\ v_{2,1} & v_{2,2} & -v_{2,3} \\ v_{3,1} & v_{3,2} & -v_{3,3} \end{bmatrix}$ and $\mathbf{X}$ with $\mathbf{X}' = \mathbf{V}'\mathbf{U}^T$.

Implementation on actual experimental data is presented in Section 4.3 . Other cases are considered in [32]. Commercial implementations for larger point sets usually employ quaternion methods [33].

2.2.4 Commercial Point Cloud Software

A number of point cloud processing software packages are commercially marketed, including GeoMagic [34], PolyWorks [35], etc. Basic software functions include point cloud viewing, point deletion, triangular facet surface creation, and CAD file import. To align the MCS point cloud to CAD coordinates, registration algorithms are required. The most basic registration methods are derived from the Iterative Closest Point (ICP) work of McKay and Besl [33]. For manufacturing operations, the use of Geometric GD&T datums and alignments is industry standard. This requires feature fitting of lines, planes, etc. and the ability to lock any combination of the six (three translational and three rotational) rigid body degrees of freedom. The above cited software has ability to accomplish this, but, being intended for interactive use, the full automation required to eliminate operator invention is lacking. For automation of the work reported herein, use of the Origin International CheckMate Engine (CME) is intended [36]. This software provides a Microsoft Windows Dynamic Link Library (DLL) set of callable functions that implement the feature fitting and registration functions for GD&T analysis.

Point Cloud Library (PCL) is a large scale, collaborative, open source software project for 2D and 3D point cloud processing. The PCL framework contains numerous state-of-the art algorithms including filtering, feature

estimation, surface reconstruction, registration, model fitting and segmentation [37]. Creation of PCL was driven by the availability of low cost, high data rate sensors such as Microsoft Kinect, which have significant potential for robotics navigation and object recognition. While it does not currently have functions relating to GD&T analysis of point clouds, given the large number of collaborators, ease of adding changes, and free for commercial and research use license, PCL is likely to experience many feature additions which could include GD&T analysis capability.

2.3 Summary

Touch trigger probes have evolved to now include higher data rate analog probes, and implementation on CNC machine tools. Very high data rate laser digitizers are now available for CMMs, but are not integrated with commercial CNC machine tool motion controllers. Commercial point cloud software deals only with standard cases, and does not interface with CNC machine based inspection methods, which is required for an in-process manufacturing implementation in a variable geometry environment. Progress towards overcoming these shortcomings is contributed in the remaining chapters of the thesis.

Chapter 3

Hardware Description

3.1 CNC Machine Tool

The work in this thesis is applicable to a range of CNC machine tools, including CNC turning centres, 3 axis milling centres, 5 axis milling centres as well as machining centres featuring both turning and milling.

The work in this thesis used a Matsuura FX-5, 3 axis, vertical machining centre controlled by a FANUC Inc. Series 15i-MA control computer.

The FX-5 was equipped with a Renishaw MP12 touch trigger probe with a repeatability of 1.0 μm (2σ) when used with the 50mm stylus.

3.2 Touch Trigger Probe Integration

Typically touch trigger probes are used in conjunction with probing cycle programs provided by the probe manufacturer. While these cycles are configurable and can be combined to create sophisticated probing routines, the proposed alternative of processing data external to the CNC control allows use of data fitting techniques such as the OLS methods described in section 2.2 The NC DPRNT command is used to print the probe hit coordinates to a Universal Machine Interface (UMI) board over the serial

connection. The UMI will then pass the data over Ethernet to a computer for processing. To suit industries where additional computer involvement is undesirable for security reasons, the UMI can be programmed to process the data directly, using the same least-squares fitting techniques.

This chapter describes the hardware and necessary implementation details that enable this tool path alignment workflow.

3.3 Hardware Overview

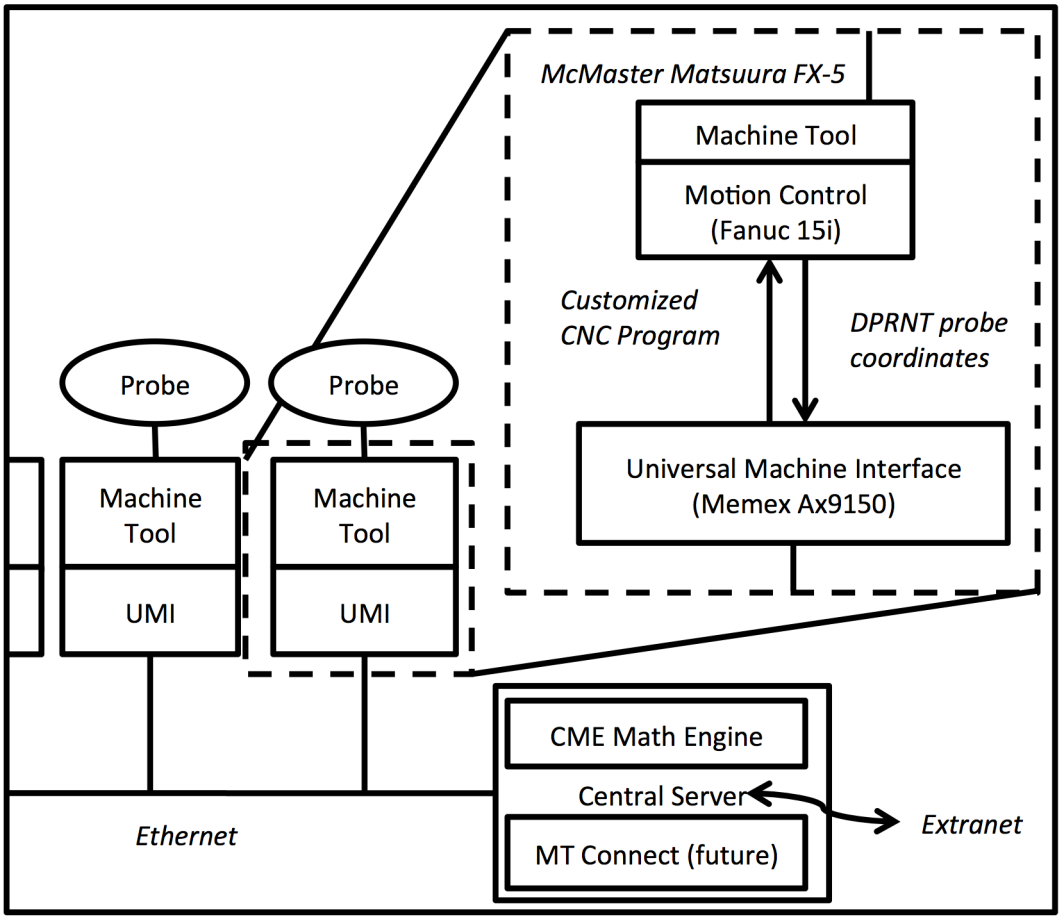


Figure 3.1 - Proposed Factory Floor Architecture

The major hardware elements are:

1. Touch Trigger Probe
2. Machine Control Computer
3. Universal Machine Interface (UMI)
4. Data processing server

A probing tool path is executed by the machine control, to obtain a measurement using the touch trigger probe. This measurement is passed to the processing computer via the UMI. Once all data points have been received, the an updated tool path is computed and sent back to the machine via the UMI. This proposed bidirectional communication is a critical step towards bringing the capabilities of GD&T software to the CNC machine tool. The following sections describe each component in greater detail.

3.4 Ax9150 UMI

The Ax9150 Universal Machine Interface developed by Memex Automation Inc. is a multi-function device originally designed to report machine status to improve overall equipment efficiency of machine tools on a factory floor. It features 0-30VDC volt digital inputs and 30VDC digital outputs for connecting to factory equipment signals, as well as RS-232 Serial and Ethernet connectivity for data reporting and transfer of part programs. Compatibility with the Fanuc 15i machine tool control, availability of manufacturer supplied software, access to local technical support, and lack of other comparable hardware were all factors in the decision to use the Ax9150 UMI.

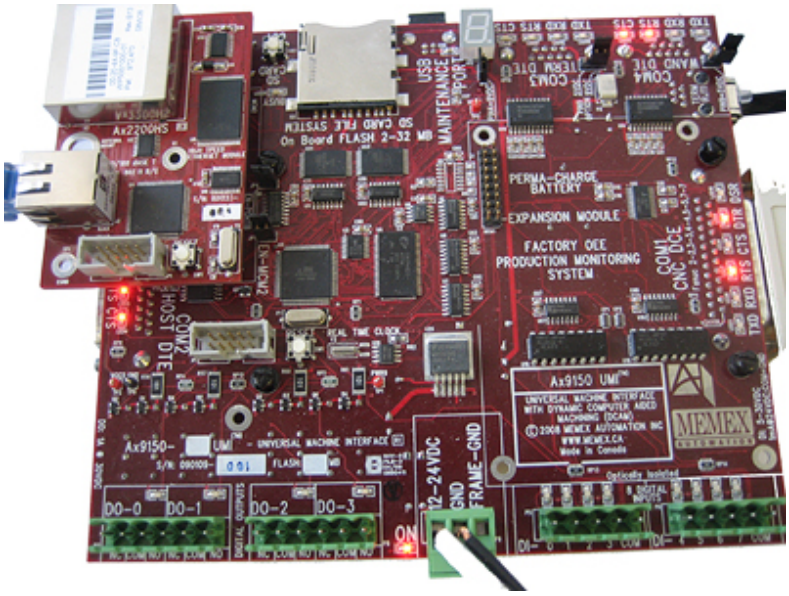


Figure 3.2 - Memex Ax9150 Universal Machine Interface [38]

3.4.1 Installation

The Ax9150 UMI was installed inside the Matsuura FX-5 Machine Tool cabinet and connected to Serial port 1 of the Fanuc Series 15i-MA control. To support the UMI Ethernet connection, a local area network router was also installed in the cabinet. The machine control system parameters #20 and #21 were changed from "1" to "14" on the CNC Controller to set the Serial port active for UMI communications.

3.4.2 Passing coordinate data to PC

In order to transfer the G31 position information from the CNC Controller, the 'Macro B' feature set of the Fanuc control was utilized. The

Macro B feature set enables a set of codes outside the RS274D G-code standard to be used in order to enable advanced programming features such as data storage, logical operations, looping and more [39]. The 'DPRNT' command was used to send plain text output to a RS-232C device, in this case the Ax9150 UMI, which then relays the data over Ethernet to the processing computer.

The following code is used to measure a point, and then print the probe coordinates stored in local variables #5061, #5062 and #5063:

```
G31X-137.Y-137.Z72.F25.
POPEN
DPRNT[POINT: #5061 [44] #5062 [44] #5063 [44]]
PCLOS
```

Resulting serial terminal output:

```
POINT: -135.5105 -136.3415 72.0620
```

3.5 Calibration of Touch Trigger Probe

Before taking a measurement the touch trigger probe must be calibrated as mounted on the CNC machine tool. This was accomplished using a ring gauge of known diameter 71.133 mm (2.8005 inches), measured using the probe in the CNC. From the resulting data, the directionally

dependent pre-travel amount is revealed. An average pre-travel of 0.062 mm was calculated and used for compensation in this work. The probing method is described in 3.5.1 and the results are discussed in 3.5.2 .

3.5.1 Calibration Tool Path

The calibration procedure involves taking probe measurements in different directions to assess the probe's directionally dependent pre-travel amount. In this procedure 36 probe measurements were made, travelling radially from the centre of the ring to the inside surface of the ring as shown in Figure 3.3.

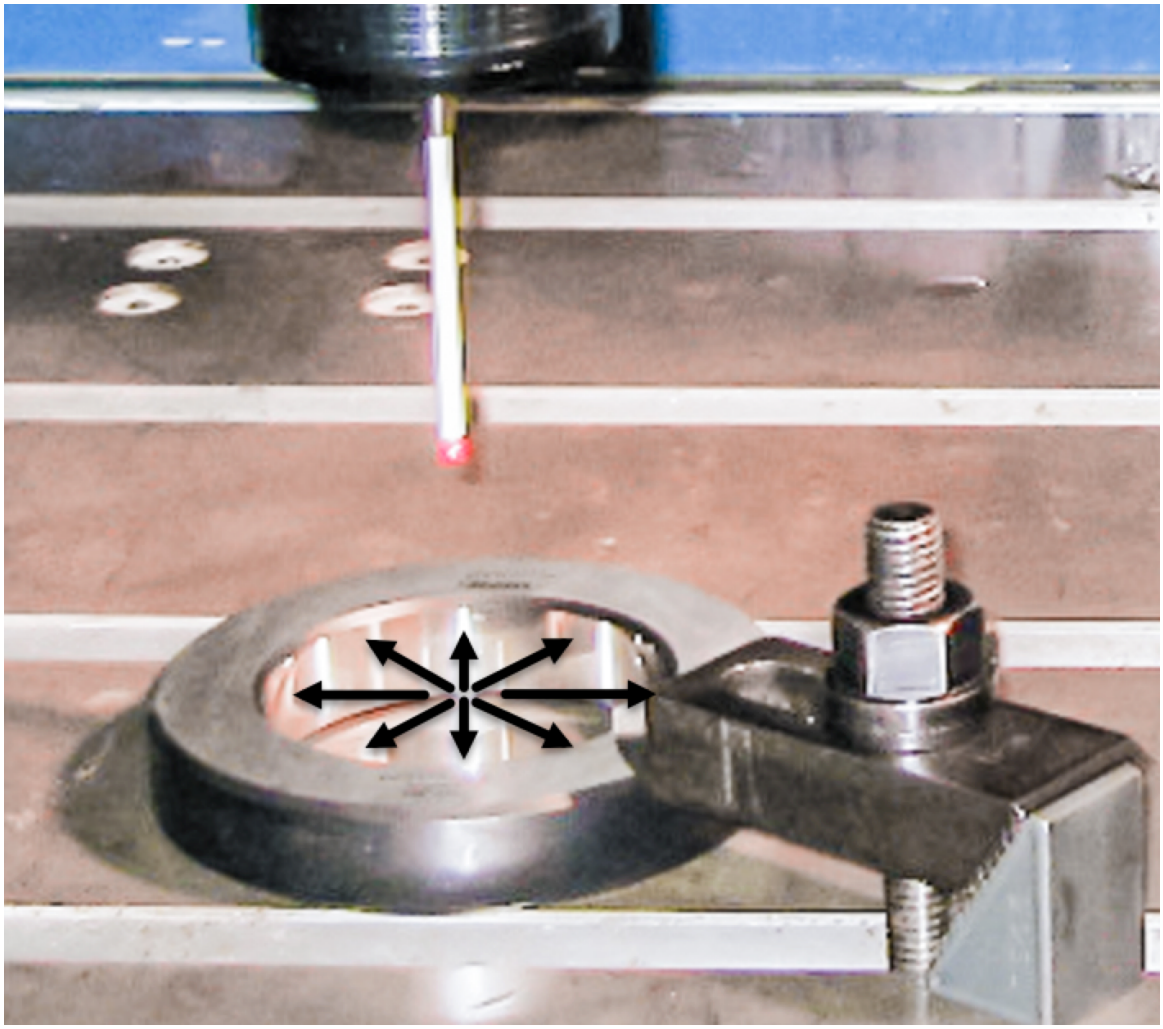


Figure 3.3 - Ring Gauge Probing Path. Arrows indicate 8 of 36 trajectories.

To reduce the error associated with the probing speed and the unpredictable 0-4ms delay between the rising edge of the skip signal and the reaction of the machine [40], a double hit method illustrated in Figure 3.4 was used. Upon making initial contact with the ring gauge surface **(1)**, the probe backs away from the contact point a small amount **(2)**. The probe

makes a second measurement of the location with reduced feedrate to limit delay effects **(3)**. The probe then moves to another inspection site **(4)**.

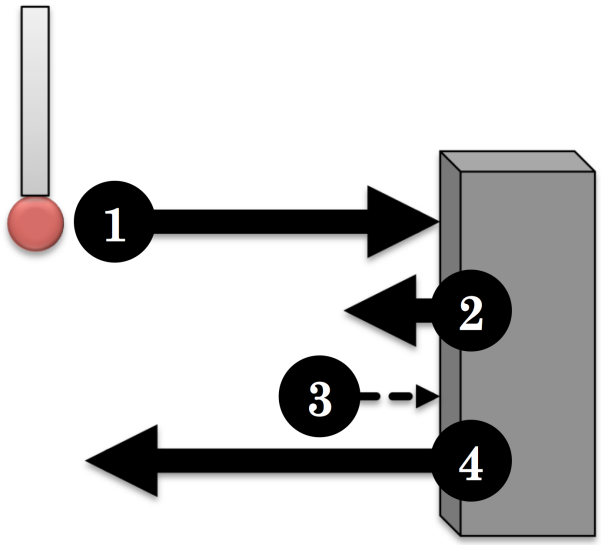


Figure 3.4 - 2 Hit Probing Method

3.5.2 Results of Calibration Data

From the measured ring gauge radius data, the average pre-travel was calculated to be 0.062 mm. The correction was applied independent of direction, however, all probe hits in later experiments were in line with the major axes (0,90,180,270) where pre-travel error is very close to 0.062 mm. Pretravel in the axial direction of the probe stylus (Z- direction) is governed by the deflection due to compressive forces, the which is much less than the deflection due to bending. Because of this, no pre-travel compensation was applied to axial direction probe hits.

Figure 3.5 shows the pre-travel amounts plotted on a circular grid. The triangular shape is characteristic of the probe's kinematic configuration – three mechanical supports with a retaining spring.

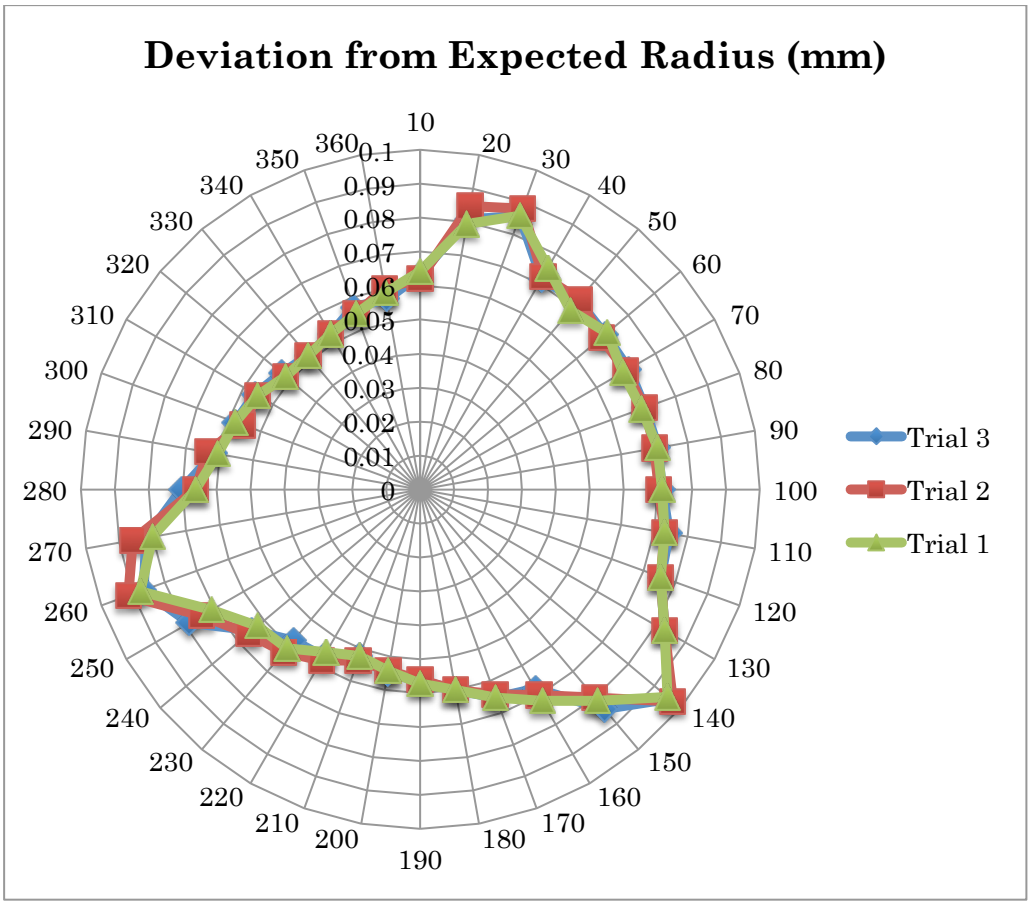


Figure 3.5 - Graph of Pre-travel variation using 36 trajectories

3.6 Three Sphere Probing Routine

A probing routine was created to obtain sphere centre data for the fixture plate illustrated in Figure 4.2. The operator positions the probe above the first sphere so that contact will be made with the top hemisphere as the

probe moves downwards. After contact is made in the downwards direction, a coarse approximation of the X-Y centre of the sphere is found by probing at 4 equally spaced locations on the circular cross section located one radius distance down from the point of first contact, as shown in Figure 3.6.

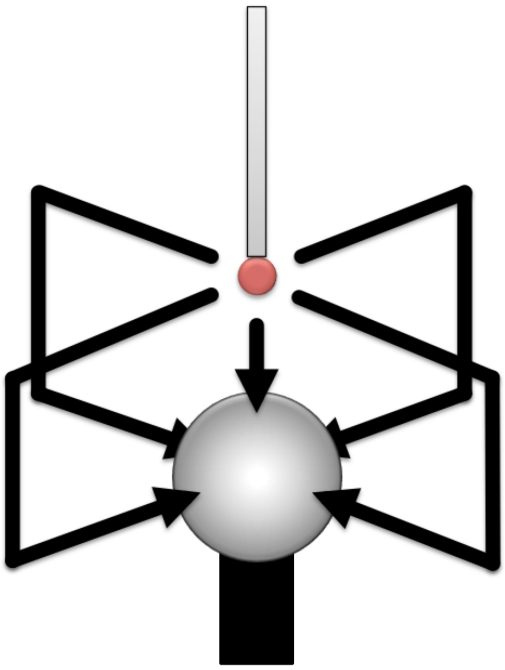


Figure 3.6 - 5 Point Probing of a Sphere

Once the centre is found, the process is repeated using this centre as a starting point. This method ensures all the final probe hits are made normal to the sphere surface to minimize pre-travel errors.

Because the sphere locations on the plate are known, the probing routine can find the top hemispheres of spheres 2 and 3 by moving in X or Y by the sphere spacing if rotation misalignment is limited. For misalignments

expected to be greater than 5 degrees, a robust seeking strategy must be applied or the datum features must be probed separately. Development of such a strategy would be specific to the workpiece and datum feature locations, and is not in the scope of this thesis.

Chapter 4

Registration Experiment

4.1 Experiment Tool Path

A partial arc of a cylindrical surface is used as the test workpiece. The machining operation is representative of a finishing operation, where the cylindrical arc surface has been roughly cut by a band saw or other non-CNC means, and CNC machining is used to create the desired surface finish. Figure 4.1 shows the toolpath trajectories in CAM software [41].

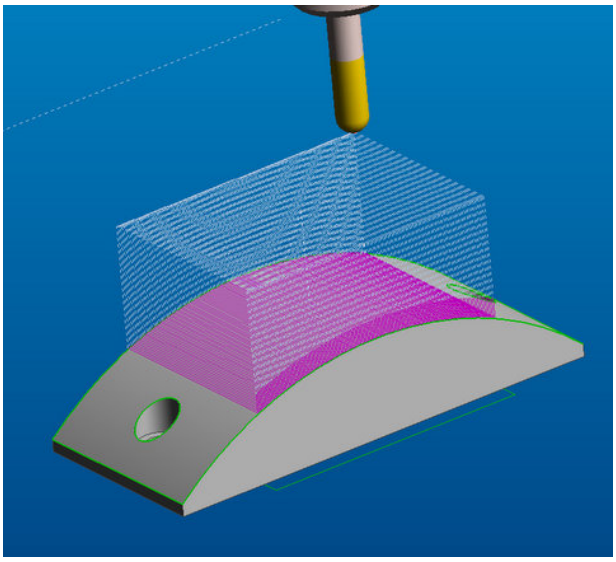


Figure 4.1 - Surface Finishing Operation Toolpath

4.2 Tool Path Adjustment

4.2.1 Part Registration in Manufacturing Processes

In high volume production, the reliability of dedicated fixture and clamping assemblies justifies the initial fixture design investment. These fixtures are created with a single workpiece in mind, and are not suited for significant workpiece variances or custom machining operations.

Herein a method is proposed whereby the clamped workpiece is located using automated sensor measurement methods, and tool path trajectories are updated at run-time to match the actual workpiece pose. In addition to reducing set-up times, scrap avoidance can be achieved by using early part inspection to detect machinability issues, such as geometric distortion in cast or heat treated parts. In these cases, analyzing the available workpiece material distribution with GD&T software gives the ability to adjust the tool path to best fit the available material, so that all features can be machined to meet the GD&T specifications for the part.

4.2.2 Tooling Sphere Registration Method

As a simple testing platform, a base or platform was constructed as illustrated in Figure 4.2.

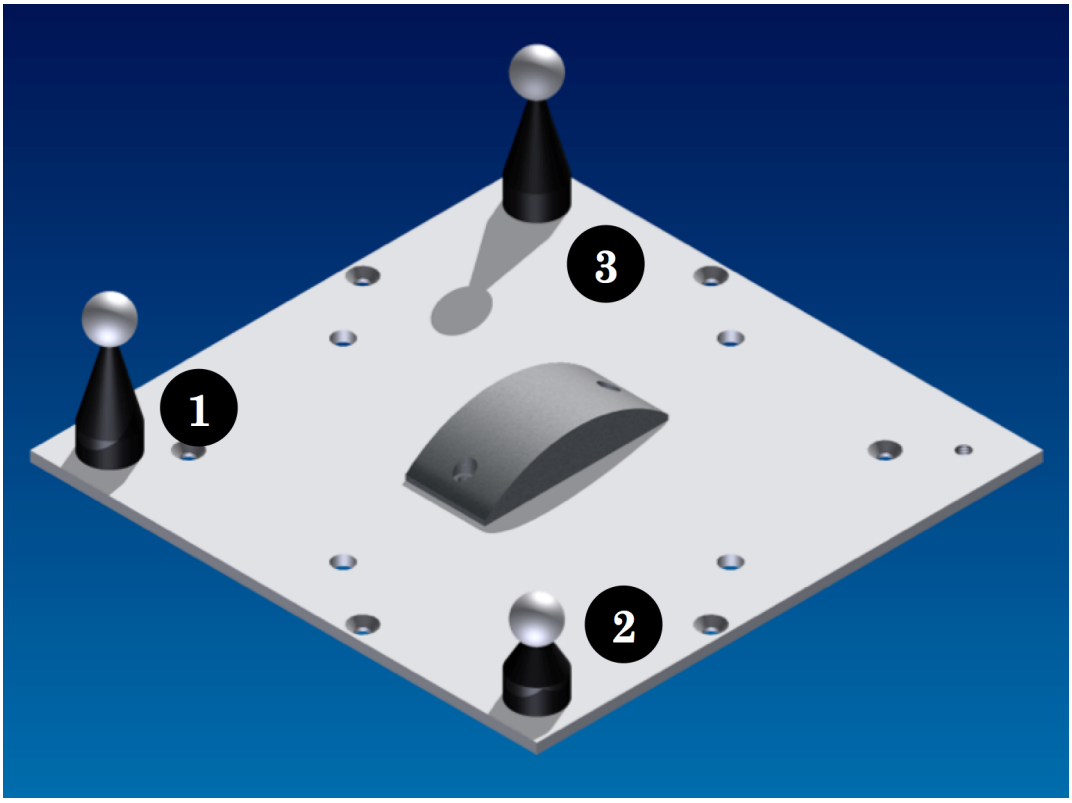


Figure 4.2 - Fixture Base with Tooling Spheres and Workpiece

The base consists of three tooling spheres fixed to a plane, on which another workpiece can be mounted. Ceramic tooling spheres were chosen to provide simple geometry for touch trigger probe inspection. The ceramic tooling spheres are also suitable for laser scanning technology due to the diffuse reflection offered by the ceramic surface.

Once the base assembly is mounted in the CNC workspace, a NC probing routine will obtain point data to solve for the centre of each of the tooling spheres. Once the point locations are known in the MCS, the HTM to

convert from CAD to MCS can be calculated as illustrated in section 2.2.3 , and the correction transformation can be applied to the nominal tool path.

Samples of original and adjusted toolpath programs can be found in Appendices E and F. A laser scan of the finished workpiece will be used to provide numerical data to evaluate the success of the proposed workflow.

4.3 Test Cases

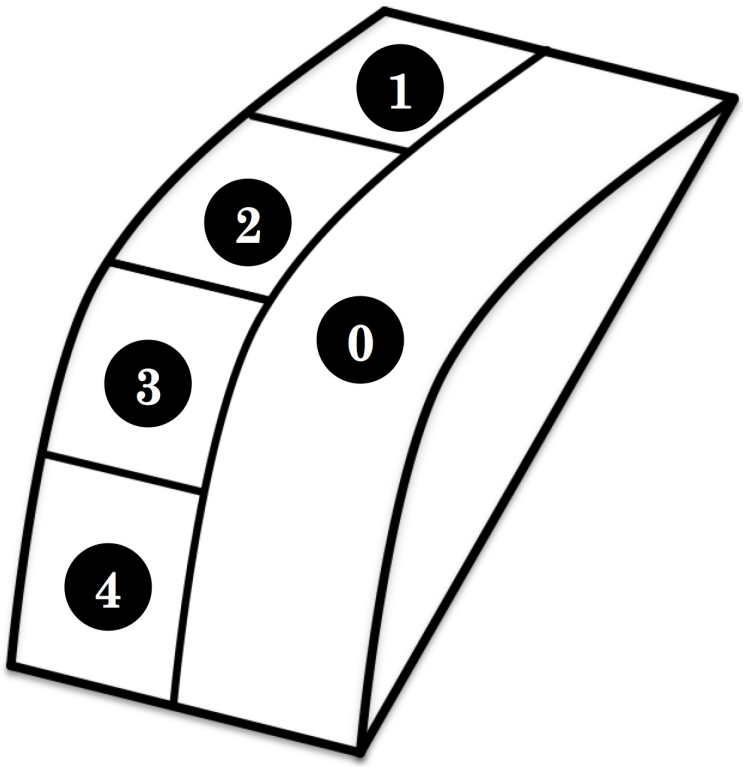


Figure 4.3 - Toolpath Case Divisions

To test the consistency of the registration, the conventional machining toolpath shown in Figure 4.1 is split into 5 sections shown in Figure 4.3. Each section will be machined after moving and re-probing the fixture, fitting probe data, and adjusting the tool paths.

In Case 0, the fixture is clamped in the nominal position with the workpiece axes aligned with the machine axes. In this position the spheres are probed and stored for reference in later registration operations.

In Case 1 the plate is rotated by 1 degree positively about the Z axis. In Case 2 the plate is rotated by -1 degree about Z, and +16 mm in X. In Case 3, the plate is rotated by 1 degree about Z, and moved -48 mm in X. And finally in Case 4, the plate is rotated -2 degrees about Z and moved -35 mm in X. All specified translations and rotations are in reference to the nominal position. The rotation in Case 4 is the maximum rotation that the plate's oversized bolt holes would permit. Machining of Case 0 on the Matsuura F-X5 3 axis CNC machine is pictured in Figure 4.4.

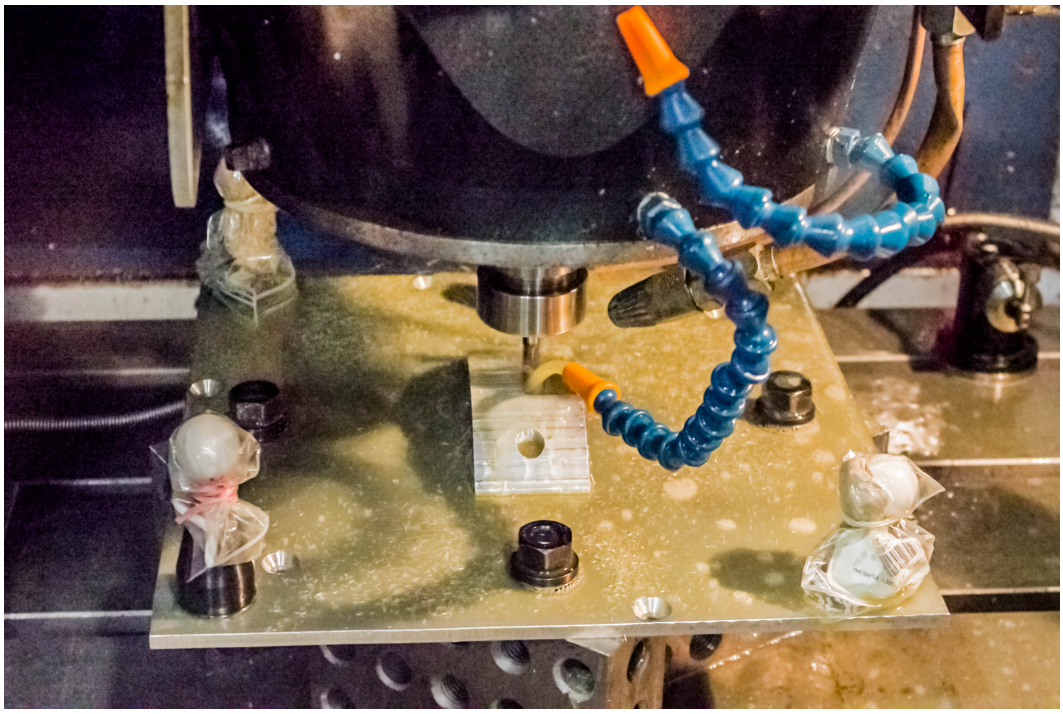


Figure 4.4 - Machining Case 0

Table 4.1 - OLS Sphere Fit

Case 0 – Sphere 3			
Point	X (mm)	Y (mm)	Z (mm)
1	-136.7110	136.4120	148.8330
2	-124.0060	136.4140	136.1330
3	-149.4150	136.4140	136.1330
4	-136.7120	149.1310	136.1330
5	-136.7120	123.6930	136.1330
Fitted Sphere Centre	X (mm)	Y (mm)	Z (mm)
	-136.7105	136.4120	136.1212
	Radius (mm)		12.7118
	Standard Deviation of Residuals (mm)		0.0145
	Uncertainty of Centre (mm)		0.0073

Table 4.2 - Fitted Sphere Centre Registration Error and Registration Components

	Fitted Sphere Centre Registration Error			Registration Components			
Case 0	X	Y	Z	R			T
Sphere 1	0.00E+00	1.71E-13	0	1.0000	0.0000	0.0000	0.00E+00
Sphere 2	5.68E-14	8.53E-14	0	0.0000	1.0000	0.0000	-1.42E-14
Sphere 3	-5.68E-14	-8.53E-14	0	0.0000	0.0000	1.0000	1.42E-14
Case 1							
Sphere 1	-0.0022	0.0148	0.0002	0.9999	-0.0173	-0.0002	-0.8700
Sphere 2	0.0070	-0.0049	-0.0006	0.0173	0.9999	-0.0000	-0.5866
Sphere 3	-0.0049	-0.0098	0.0005	0.0002	0.0000	1.0000	0.0060
Case 2							
Sphere 1	0.0011	0.0053	-0.0001	0.9999	0.0157	-0.0001	16.6543
Sphere 2	0.0085	-0.0098	-0.0008	-0.0157	0.9999	0.0000	-0.6807
Sphere 3	-0.0096	0.0045	0.0009	0.0001	0.0000	1.0000	0.0117
Case 3							
Sphere 1	-0.0114	0.0029	0.0011	1.0000	-0.0062	-0.0002	-48.4372
Sphere 2	0.0182	-0.0064	-0.0017	0.0062	1.0000	0.0000	-1.9574
Sphere 3	-0.0067	0.0036	0.0006	0.0002	0.0000	1.0000	0.0219
Case 4							
Sphere 1	-0.0113	0.0046	0.0011	0.9993	0.0377	0.0000	-35.1846
Sphere 2	0.0087	0.0026	-0.0008	-0.0377	0.9993	0.0002	-0.2768
Sphere 3	0.0026	-0.0072	-0.0003	-0.0000	-0.0002	1.0000	-0.0090

Table 4.2 shows the 3 point registration sphere centre errors, for Cases 0 through 4 using the OLS fit method described in [32].

Chapter 5

Registration Experiment Results

5.1 Laser Scan of Finished Workpiece

After machining, the part was removed from the fixture plate and scanned using a Roland PICZA LPX-600 laser scanner with accuracy ±0.050 mm given by the manufacturer [42]. The 3 minute scan captured 160,000 points, spaced at the scanner minimum of 1 mm.

The scan data of Case 0 was fit with the nominal cylinder arc geometry and a deviation report was generated, comparing scan data from Cases 1 through 4 with the nominal cylinder arc geometry. Deviation from nominal is illustrated by the colour of the surface, as shown in Figure 5.3.

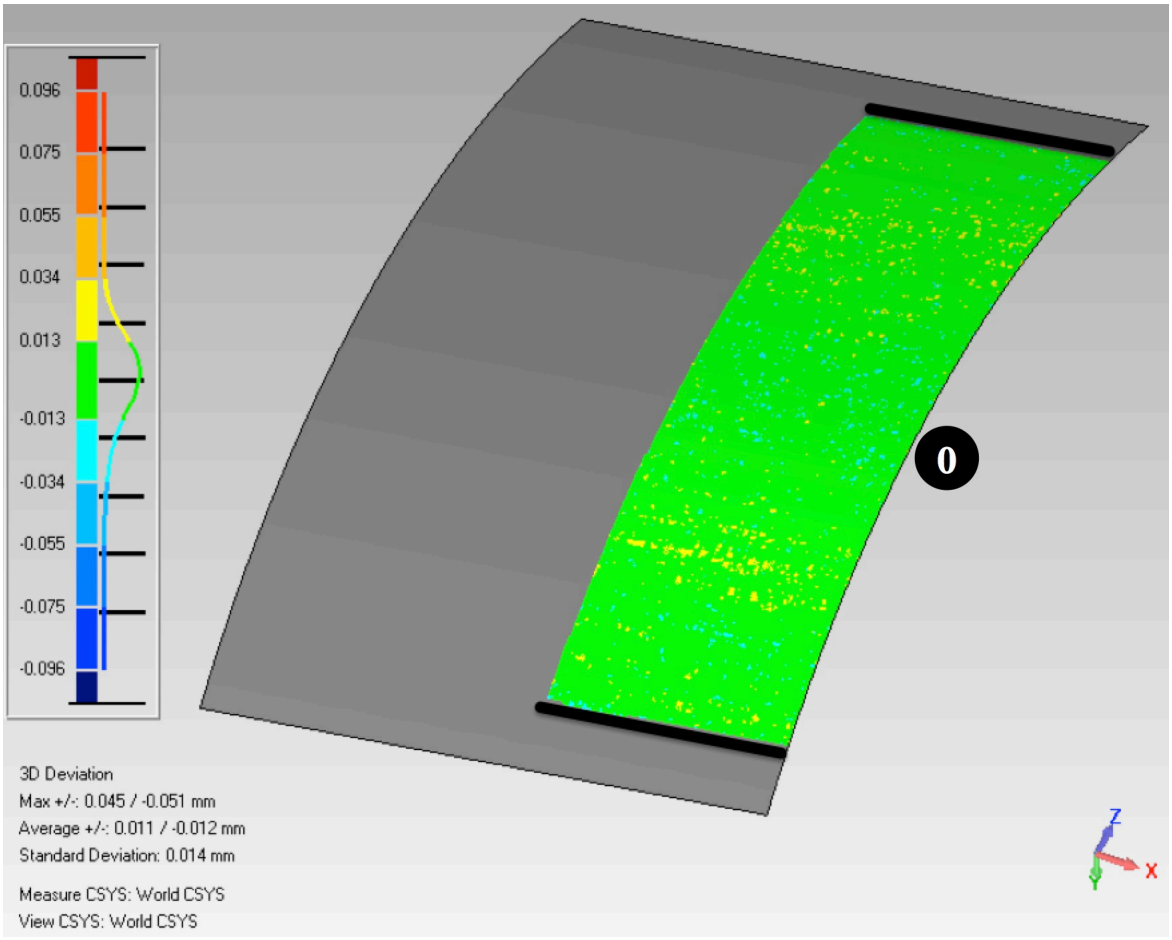


Figure 5.1 - Inspection Report of Experiment Case 0, Roland Scan Data

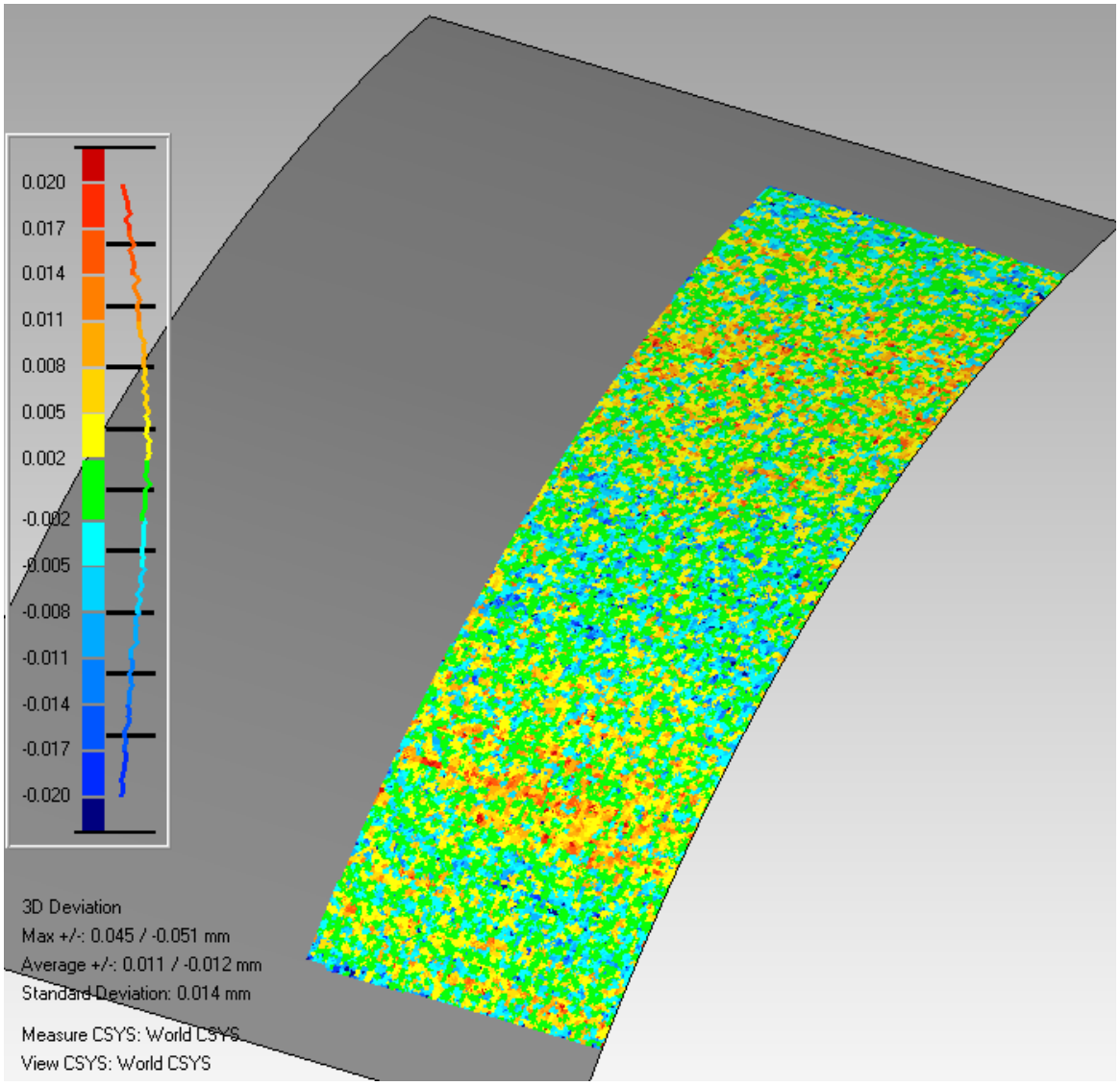


Figure 5.2 - Case 0, Roland Scan Data – Colour Map adjusted to show detail.

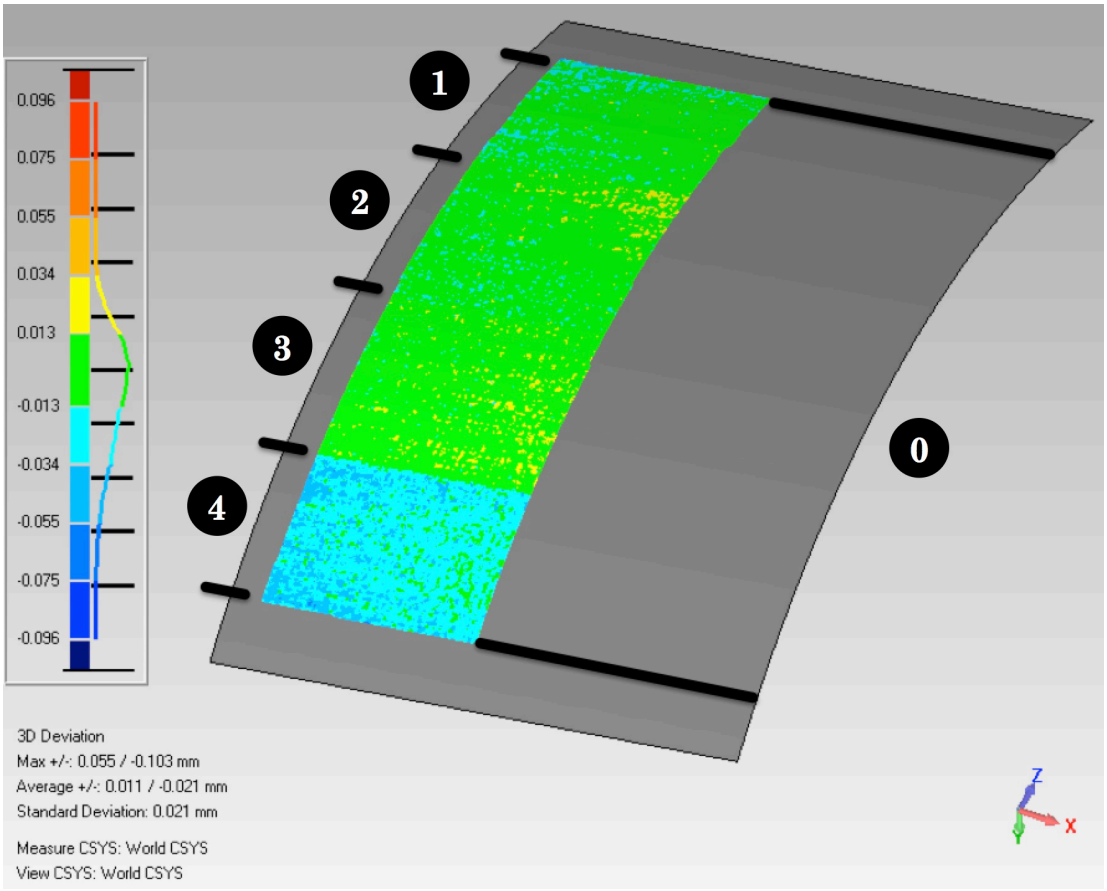


Figure 5.3 - Inspection Report of Experiment Cases 1-4, Roland Scan Data

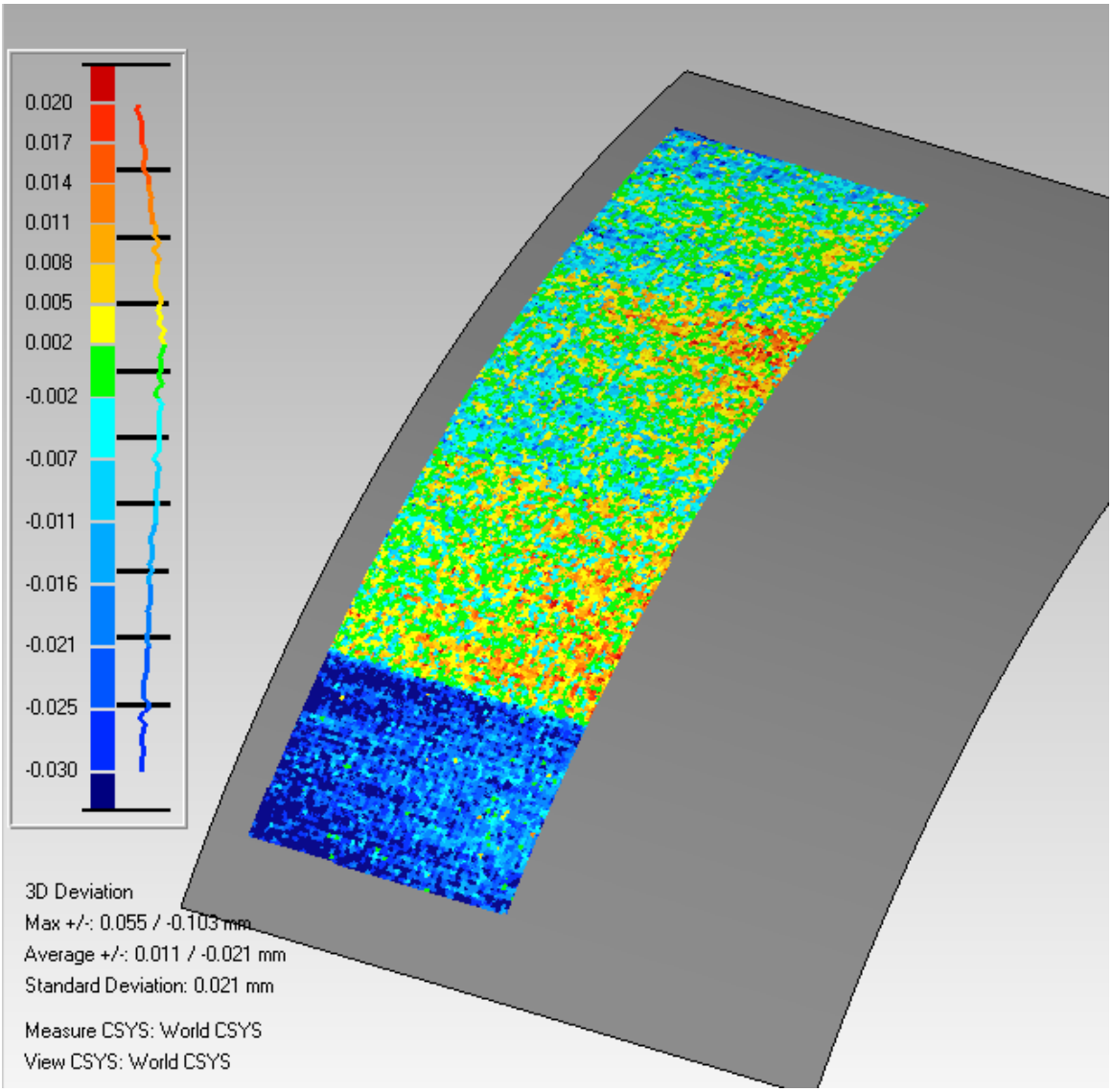


Figure 5.4 – Cases 1-4, Roland Scan Data – Colour Map adjusted to show detail.

Table 5.1 - Experiment Inspection Data (Roland) Statistics

Average	+0.011mm	-0.021mm
Standard Deviation	0.021 mm	

These results are encouraging. During their work on surface flank milling of jet engine and compressor airfoils, Wu observed that deviations in the range of 0.075 mm to 1.7 mm to the optimal airfoil shape as generated by Computational Fluid Dynamics software, did not cause noticeable losses in performance tests [43]. From the standard deviation of 0.021 mm, the system has a 98.26% ability to achieve the DIN ISO 2768 f (fine) class tolerance of ±0.05 mm for the standard's smallest nominal linear dimension category, 0.5-3 mm.

Table 5.2 - Sphere Fit Statistics

	Sphere	Fitted Radius (mm)	Standard Deviation of Residuals (mm)	Uncertainty of Centre (mm)
Case 0	Sphere 1	12.7130	0.0150	0.0075
	Sphere 2	12.7130	0.0150	0.0075
	Sphere 3	12.7118	0.0145	0.0073
Case 1	Sphere 1	12.7153	0.0065	0.0033
	Sphere 2	12.7140	0.0130	0.0065
	Sphere 3	12.7133	0.0145	0.0073
Case 2	Sphere 1	12.7125	0.0130	0.0065
	Sphere 2	12.7145	0.0150	0.0075
	Sphere 3	12.7128	0.0155	0.0078
Case 3	Sphere 1	12.7160	0.0100	0.0050
	Sphere 2	12.7143	0.0155	0.0078
	Sphere 3	12.7113	0.0155	0.0078
Case 4	Sphere 1	12.7133	0.0125	0.0063
	Sphere 2	12.7160	0.0130	0.0065
	Sphere 3	12.7130	0.0130	0.0065
Average		**12.7136**	**0.0134**	**0.0067**

Additional analysis of experimental data was performed in order to trace the errors introduced by each step. First, the OLS sphere fit statistics are analyzed, showing an average residual of 0.0134 mm, and an average uncertainty of the centre location of 0.0067 mm. The complete dataset is given in Table 5.2.

Table 5.3 - Sphere Centre Distances

		Distance (mm)		Difference from Case 0 (mm)	
		Sphere 1	Sphere 2	Sphere 1	Sphere 2
Case 0	Sphere 2	273.0136		0.0000	
	Sphere 3	273.0408	385.9398	0.0000	0.0000
Case 1	Sphere 2	273.0094		-0.0042	
	Sphere 3	273.0653	385.9373	0.0245	-0.0025
Case 2	Sphere 2	273.0097		-0.0039	
	Sphere 3	273.0416	385.9195	0.0009	-0.0203
Case 3	Sphere 2	272.9890		-0.0246	
	Sphere 3	273.0402	385.9184	-0.0006	-0.0214
Case 4	Sphere 2	272.9935		-0.0201	
	Sphere 3	273.0522	385.9466	0.0115	0.0068

A simple metric to check for warpage in the plate is the distance between the sphere centres. Table 5.3 gives the centre-to-centre distances calculated between each sphere for each case. To compare the distances for each case, the difference from Case 0 was also computed. The data does not suggest that any significant amount of was present, but Table 5.4 gives a better analysis using an OLS plane fit of probe data obtained from the plate surface.

Table 5.4 - Plane Fitting Data

		Case 0	Case 1	Case 2	Case 3	Case 4
Centroid (mm)	X	0.5890	3.2840	18.6880	-46.8100	-33.5140
	Y	1.5120	1.9350	-6.5650	-6.8000	-7.3950
	Z	63.9985	63.9987	63.9989	64.0005	64.0017
Direction Cosines	X	0.0000	-0.0001	-0.0001	-0.0001	-0.0001
	Y	0.0000	0.0000	0.0000	0.0001	0.0001
	Z	1.0000	1.0000	1.000	1.0000	1.0000
Residual Standard Deviation (mm)		0.0160	0.0167	0.0175	0.0189	0.0182

The difference between the minimum (Case 0) and maximum (Case 4) centroid Z height is 3.2 μm. For each case, direction cosines describe the plane to be effectively normal to the Z axis. The standard deviation of the data fit residuals indicates the plate surface flatness was consistent between trials. A photograph of the machined workpiece is presented in Figure 5.5.

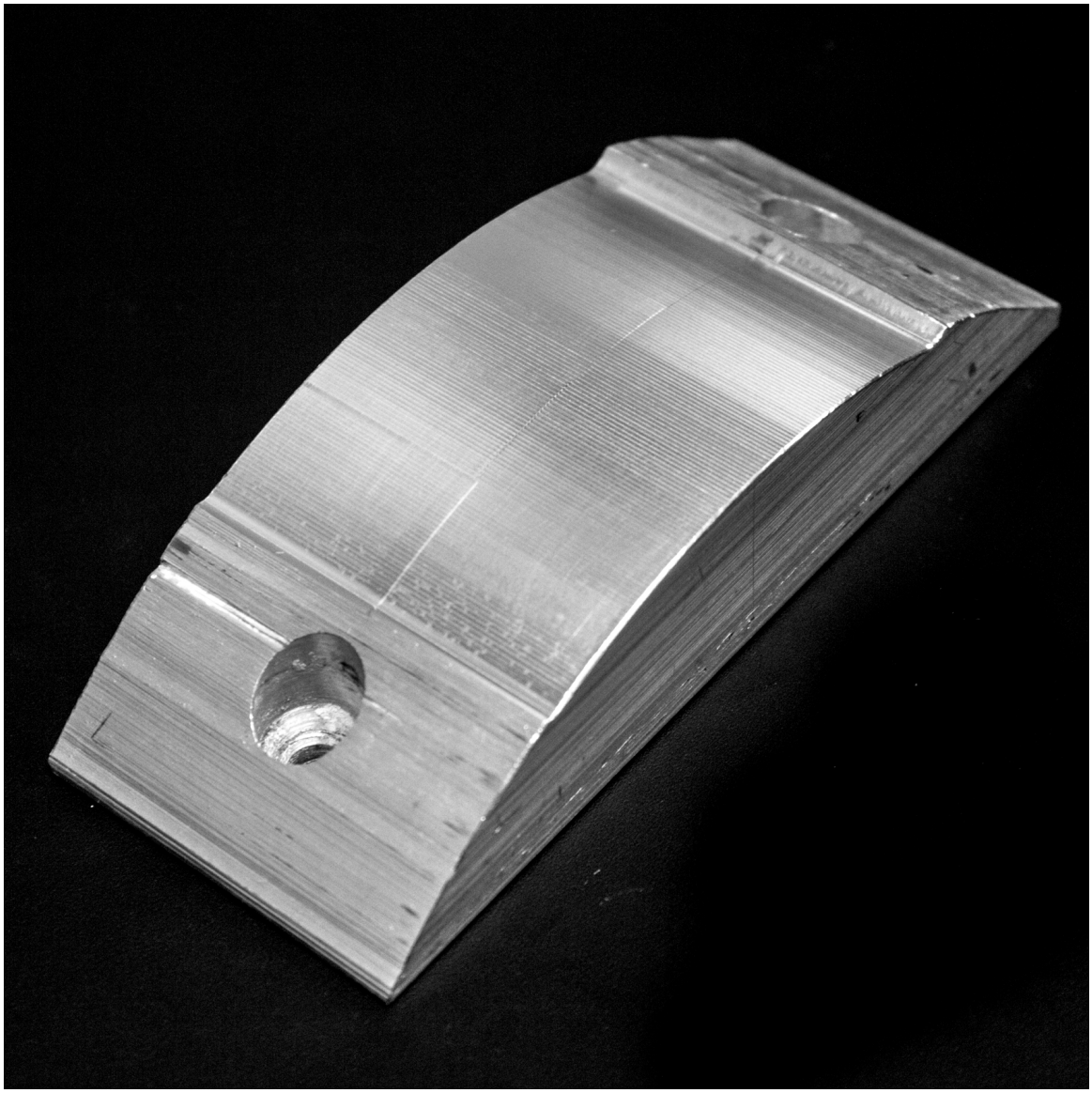


Figure 5.5 - Photograph of Machined Experiment Workpiece

5.2 Analysis of Expected Errors

The following components of the machining experiment can be expected to contribute error:

1. Positional Accuracy of the CNC machine during measurement
2. Probe Accuracy during measurement
3. Positional Accuracy of the CNC machine during machining

Distortion of the workpiece induced by clamping or machining forces can be avoided through the use of a fixture designed to provide support at critical areas of the part. If a fixture is not available, special care must be taken to ensure clamping forces do not induce distortion. In the experiment detailed in this thesis, 2x4x6 blocks were used to support the entire bottom of the fixture plate. Additionally, the mating surfaces of the machine and the workpiece were cleaned to ensure the surfaces were free of chips or other debris that could cause workpiece distortion or damage to either surface.

The positional accuracy of the CNC used during measurement is expected to be repeatable after homing. The machine used in this experiment had a digital read out with resolution of 1.0 μm. Occasional malfunction issues were experienced when homing the Z axis. In one instance the machine homed to a location 0.180mm away from the previous day's home location, prompting a restart of the experiment.

The probe accuracy is 1.0 µm when used with the 50 mm stylus. Calibration using 36 equally spaced probing vectors, including the probing vector directions used in the experiment, was done using a ring gauge and pre-travel was found to be 67 µm for the +/- X and Y probing directions. Because a single calibration value is used the pre-travel variation may contribute an error of up to 10.0 µm.

The results of the OLS sphere centre calculation are used in an OLS data set fit to compute the HTM used for Toolpath adjustment. The HTM is calculated to 4 decimal places, making the error contribution due to rounding negligible.

The accuracy of the machine tool is a factor again during machining. The magnitude and effect of each of the expected errors is summarized in Table 5.5.

Table 5.5 - Summary of Expected Errors

Error	Magnitude (2σ)
CNC Position During Inspection	1.0 µm
Probe Repeatability	1.0 µm
Pre-Travel Compensated Probe Accuracy	10.0 µm
CNC Position During Machining	1.0 µm
Total Expected	**13.0µm**

Chapter 6

Laser Scanner Integration

For soft or fragile objects, the contact pressure from touch-based digitizers can be enough distort or destroy the object. Also, the speed of touch-based digitizers is limited by kinematic and dynamic constraints. Laser scanning technology uses the known properties of light to establish information about a surface. Advancements in image sensor and image processing technology have enabled the design of digitizers that can measure with very high speed and accuracy.

The Gocator 2300 scan head used in this chapter was purchased from LMI Technologies Inc. The scanner uses the triangulation principle shown in Figure 2.9 implemented using a laser stripe and 2D image sensor. The measurement specifications given by the manufacturer are given in Table 6.1. The definition of clearance distance, measurement range, near field of view and far field of view are shown in Figure 6.1.

Table 6.1 - Gocator 2330 Laser Scanner Specifications

Resolution Z (mm)	0.006-0.014
Resolution X (mm)	0.044-0.075
Clearance Distance (mm)	90
Measurement Range (mm)	80
Field of View (FOV) (mm)	47-85

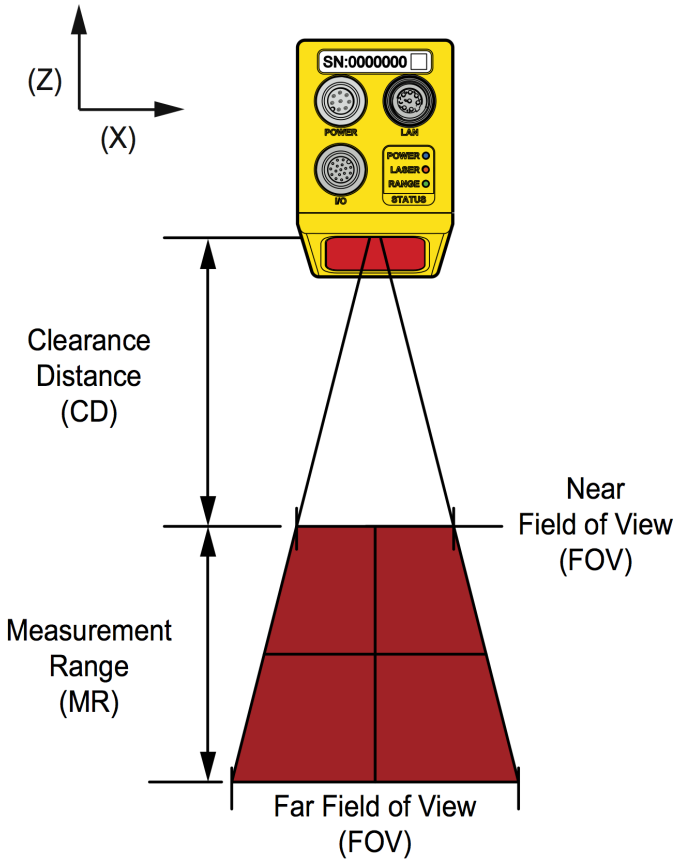


Figure 6.1 - Gocator 2330 Specifications Diagram [44]

A webserver contained in the scan head hosts a configuration web page, and enables data transmission over Ethernet.

6.1 LMI Scan Head Integration

Since measurements are only made at the intersection of the laser line and the part surface, the sensor must be attached to a movable platform in order to obtain measurements of the entire part. The platform must be able to report the exact location of the scan head so that the data obtained by the scan head can be registered to a single, world coordinate system. The following subchapters describe how the sensor was integrated with the IOTA-P Coordinate Measuring Machine (CMM) located at McMaster University. Three Renishaw model RGH22-X10D00 optical linear encoders installed on the CMM were used to provide location resolution to 1.0μm. The linear encoder output is captured by a PC based Measurement Computing Corporation PCI-QUAD04 Quadrature Counter card. During operation, a custom written PC application matches the encoder location information with the collected scan head point data.

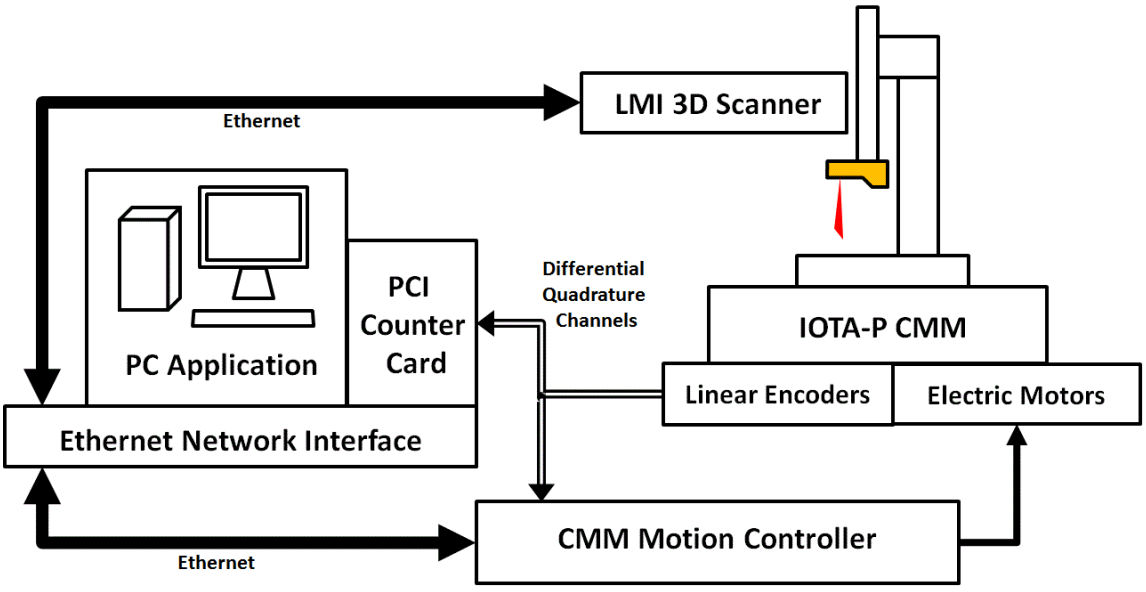


Figure 6.2 - CMM-Laser Scanner Hardware Architecture Diagram

6.2 Mount Design

A sensor mounting bracket designed and machine to be compatible with a mounting system for previous equipment (Appendix B). The mount allows the sensor attach to the z-axis and to rotate about the y-axis. A bolt-hole pattern and locking pin allows the sensor to be angled at 15 degree increments. To extend the mounting system to a CNC machining centre, a tool holder is required to attach the sensor to the spindle, and the spindle orientation would be locked using the M19 code, as is customary when using a touch-trigger probe on a CNC milling machine.

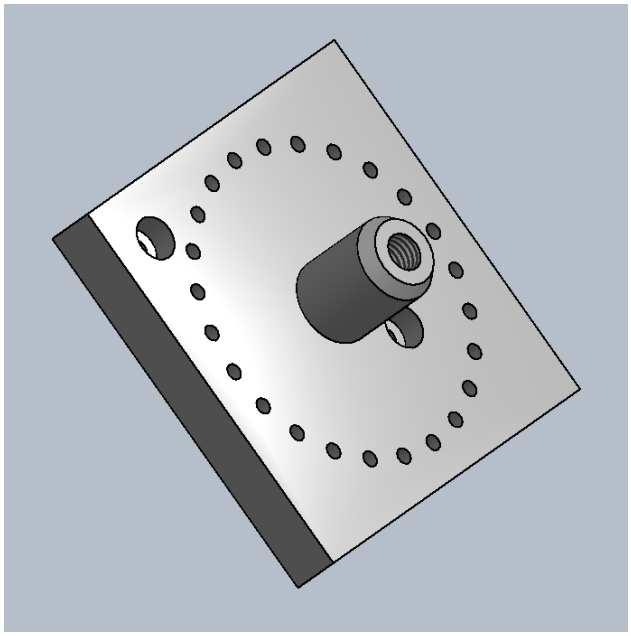


Figure 6.3 - 3D Scanner Mount with bolt hole pattern for locking pin

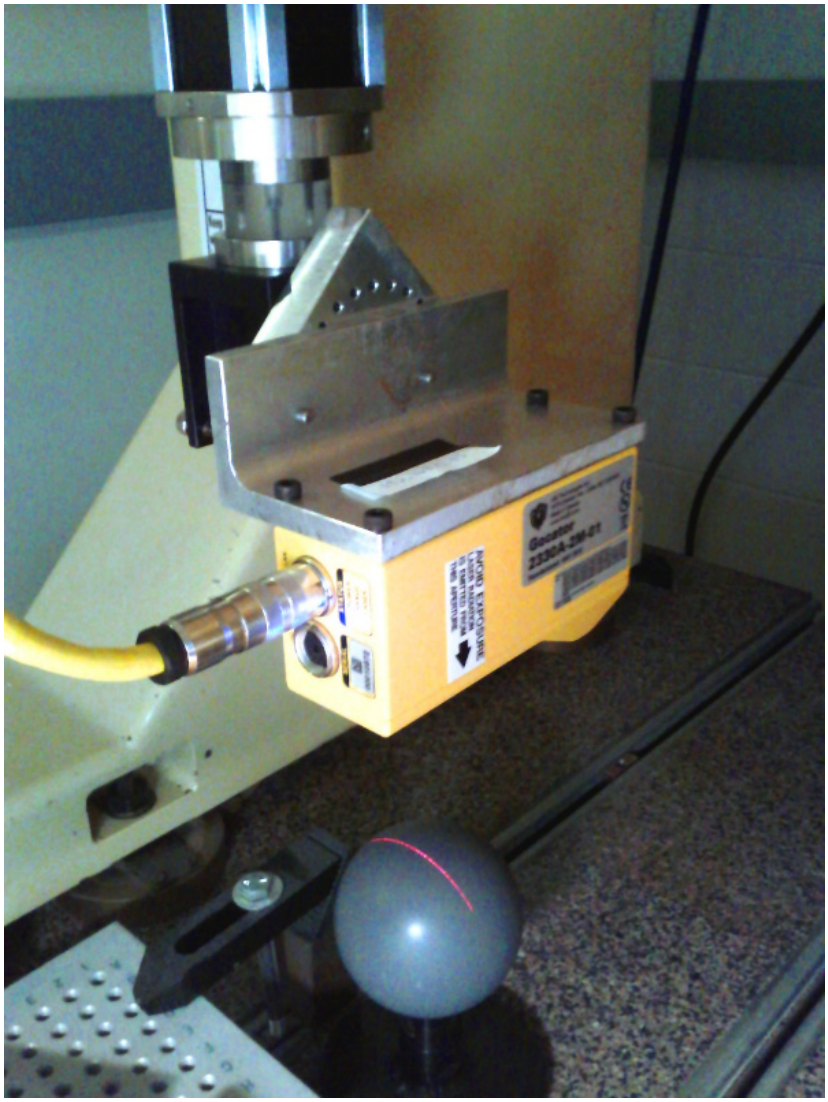


Figure 6.4 – LMI 3D Gocator 2300 Scanner on an IOTA P CMM

6.3 Desktop Control Software

The developed control software performs three functions. First, the motion of the scan head is commanded by connecting to the CMM motion control computer over Ethernet. This allows homing, and linear move

commands to be issued to the motion controller remotely. In a CNC machining setup, the motion commands must exist in a NC program.

The second function performed by the desktop application is the reading of quadrature counter card memory values, containing the axes position information.

The third and final function performed by the desktop application is commanding of the laser scan head functions. The scanner settings were configured to wait for a trigger command from the computer application before capturing a 3D profile.

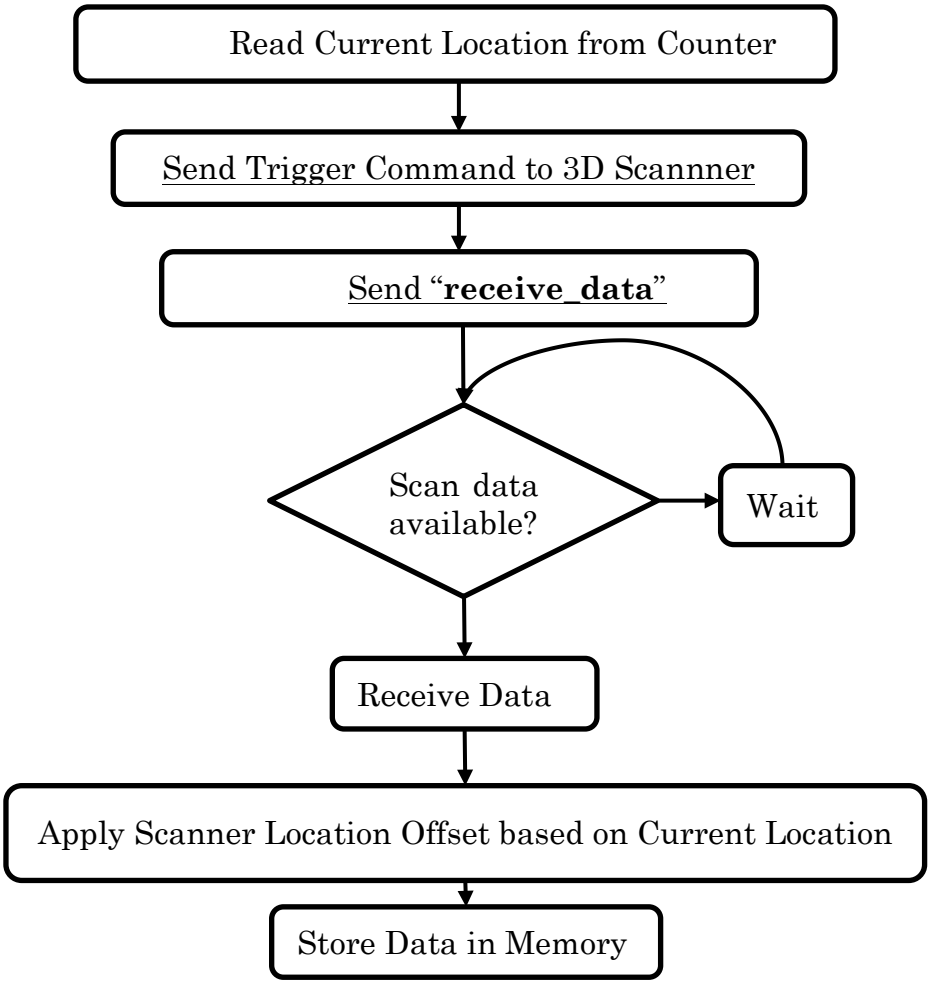


Figure 6.5 – Flow Chart of PC Application Scanning Function

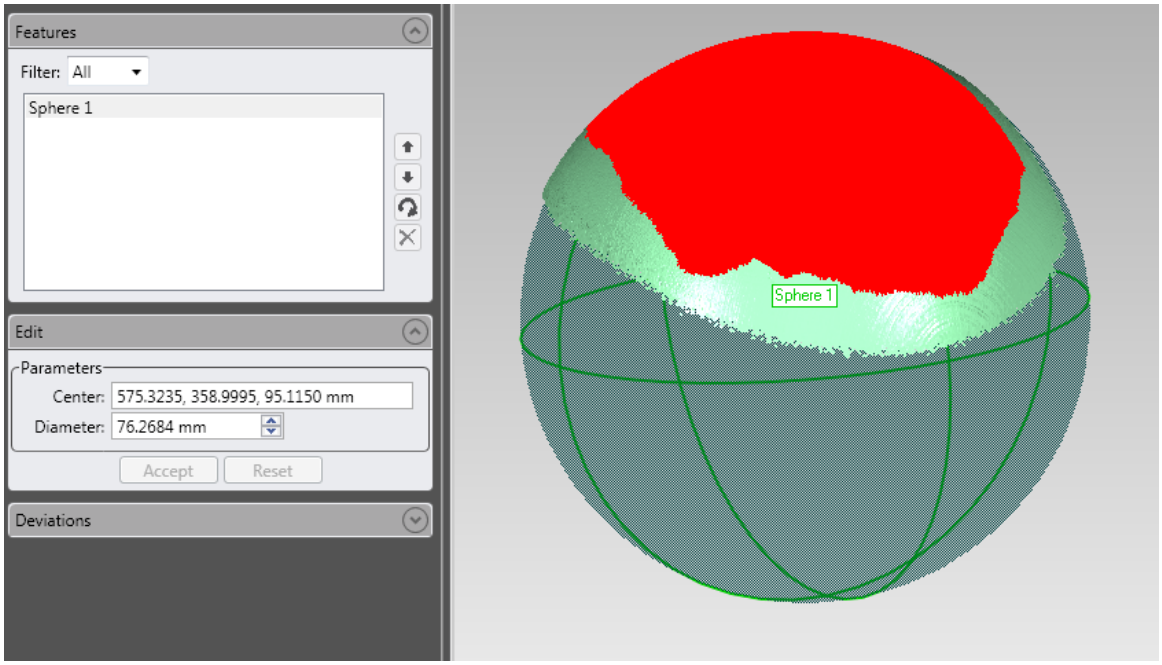


Figure 6.6 - Geometric Fit of Captured Point Cloud with 0.0684 mm error in diameter.

After homing the CMM and the initialization of the scan head, the scan can be started by clicking a button on the user interface. The application begins by reading the current location from the counter memory registers, sending the trigger command to the scan head, and then the "**receive_data**" command. Data is received into allocated memory where the scanner location offset (or rigid transformation) is applied.

Capturing a single frame of data consists of 1280 points, and is performed at 200Hz. Faster speeds are possible by reducing sample density

and through optimal lighting conditions in order to minimize the camera exposure time.

The captured data is stored uncompressed, in comma separated variable format for importing by any point cloud viewing software.

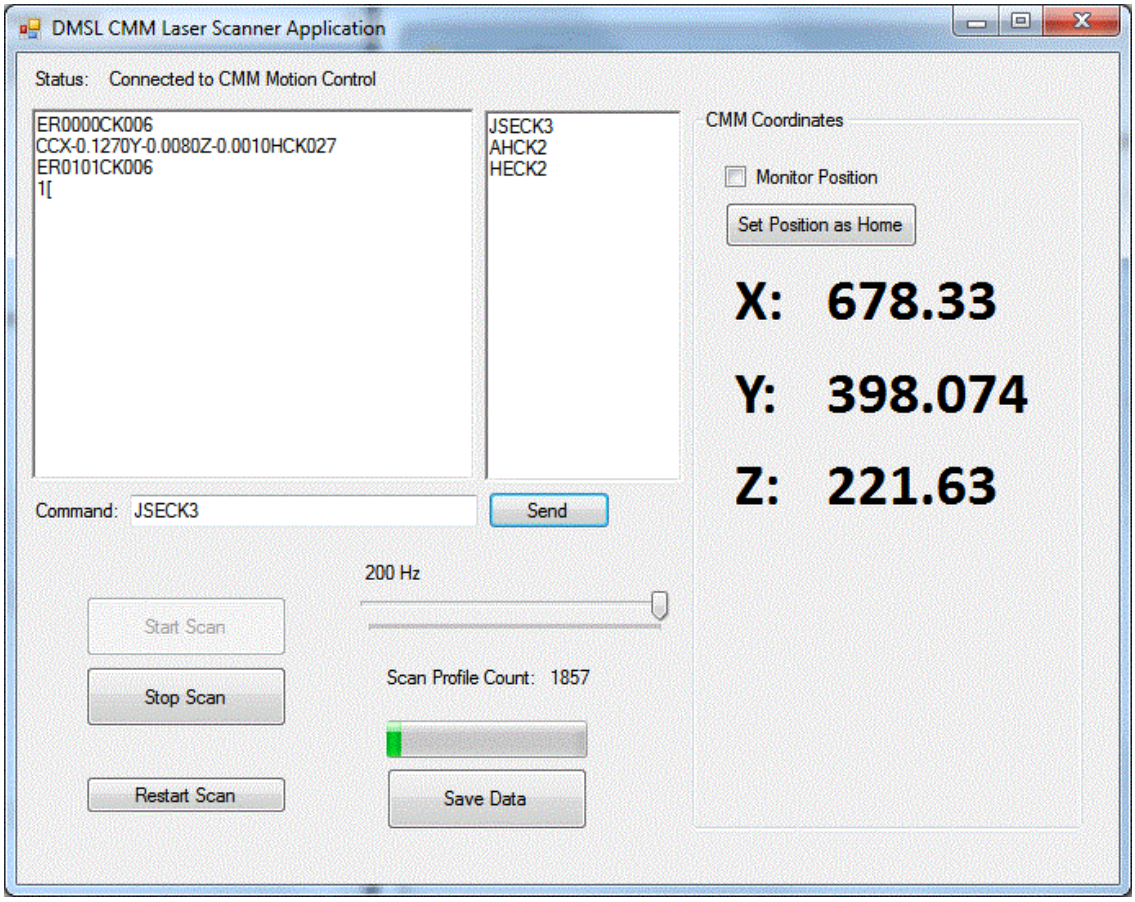


Figure 6.7 – CMM Laser Scanner Application Graphical User Interface

6.4 Expected Errors

The errors in the laser scanning system can be attributed to the following factors:

1. Positional error of the CMM due to thermal distortion
2. Positional error of the CMM linear scales
3. Delay between capture of positional data and capture of scan frame
4. Accuracy of laser scanner

The positional error of the CMM due to thermal distortion was not considered in these experiments. The small scanning distance and short time span reduce the effects of these errors.

The positional accuracy of the CMM linear scales given by the manufacturer is 1.0 μm.

Time delay between the capture of the position and the scan data creates a distance error equal to the scanning head velocity multiplied by the delay. The scanning velocity was approximately 2.5mm/second, and the scanning period of 0.005s limits the maximum distance error to 12.5 μm in the Y direction. This error can be accounted for and corrected if the time delay and velocity are known at the time of scanning.

The resolution of the laser scanner is given as 0.006-0.014mm in Z, and 0.044-0.075mm in X. Since the scan line X direction is along the part cylinder axis, only the Z scanner error will be a factor in the surface deviation report.

The magnitude of the expected final error is presented in Table 6.2.

Table 6.2 - CMM Laser Cumulative Error

Error	**Magnitude**	**Surface Scan Error (Z)**
CMM Linear Scale	1.0 µm	1.0 µm
Time Delay	12.5 µm in X	Negligible
Laser Scanner	0.006-0.014 mm in Z,	0.014mm
Total Error		**Up to 15 µm**

6.5 Test Data

The resulting workpiece from the machining experiment in Chapter 4 was scanned using the assembled laser scanner and CMM system. The 30 second scan captured 200 frames per second, each generating 1280 points for a total point count of 7.68 million points. The point spacing in the X direction of the scan head was approximately 62.5 µm, and the point spacing in the Y direction was approximately 0.127 mm.

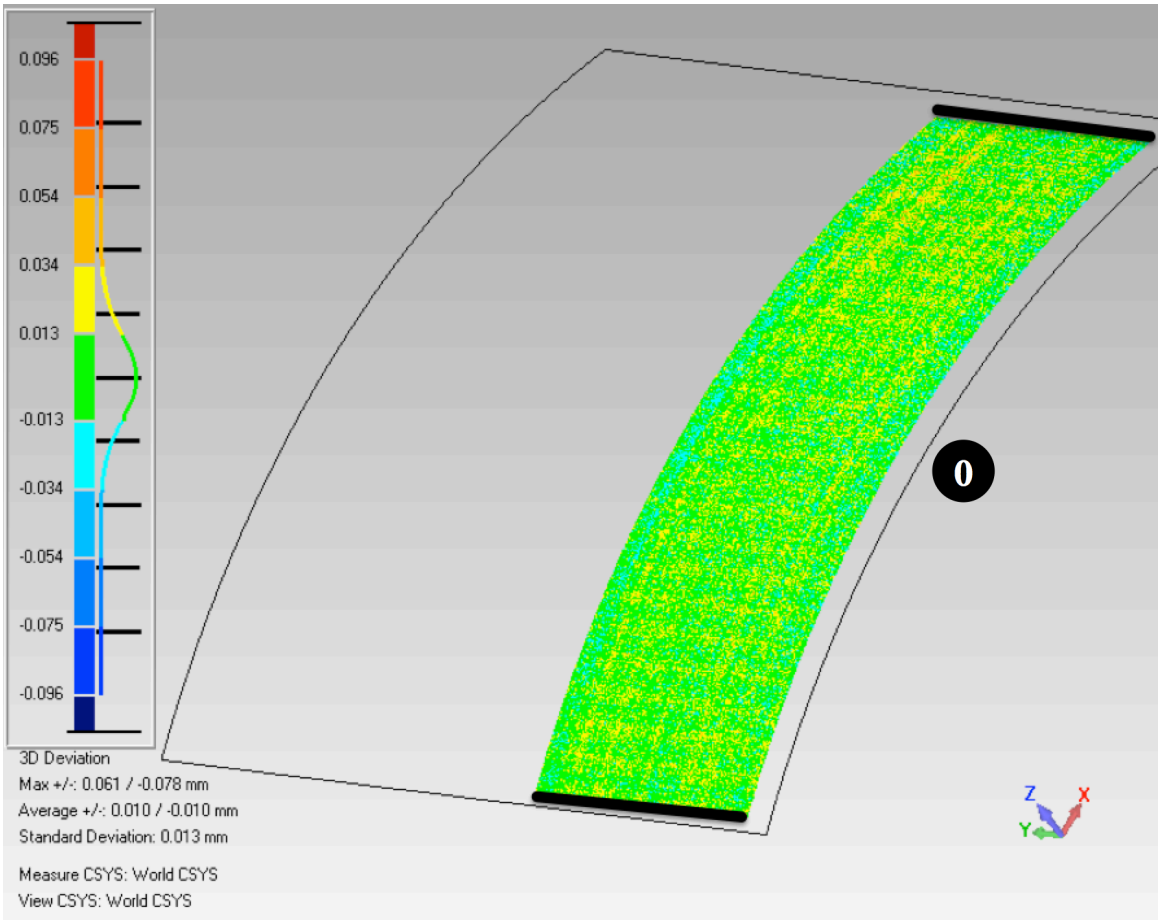


Figure 6.8 - LMI Scan Data: Case 0 Surface Deviation Analysis

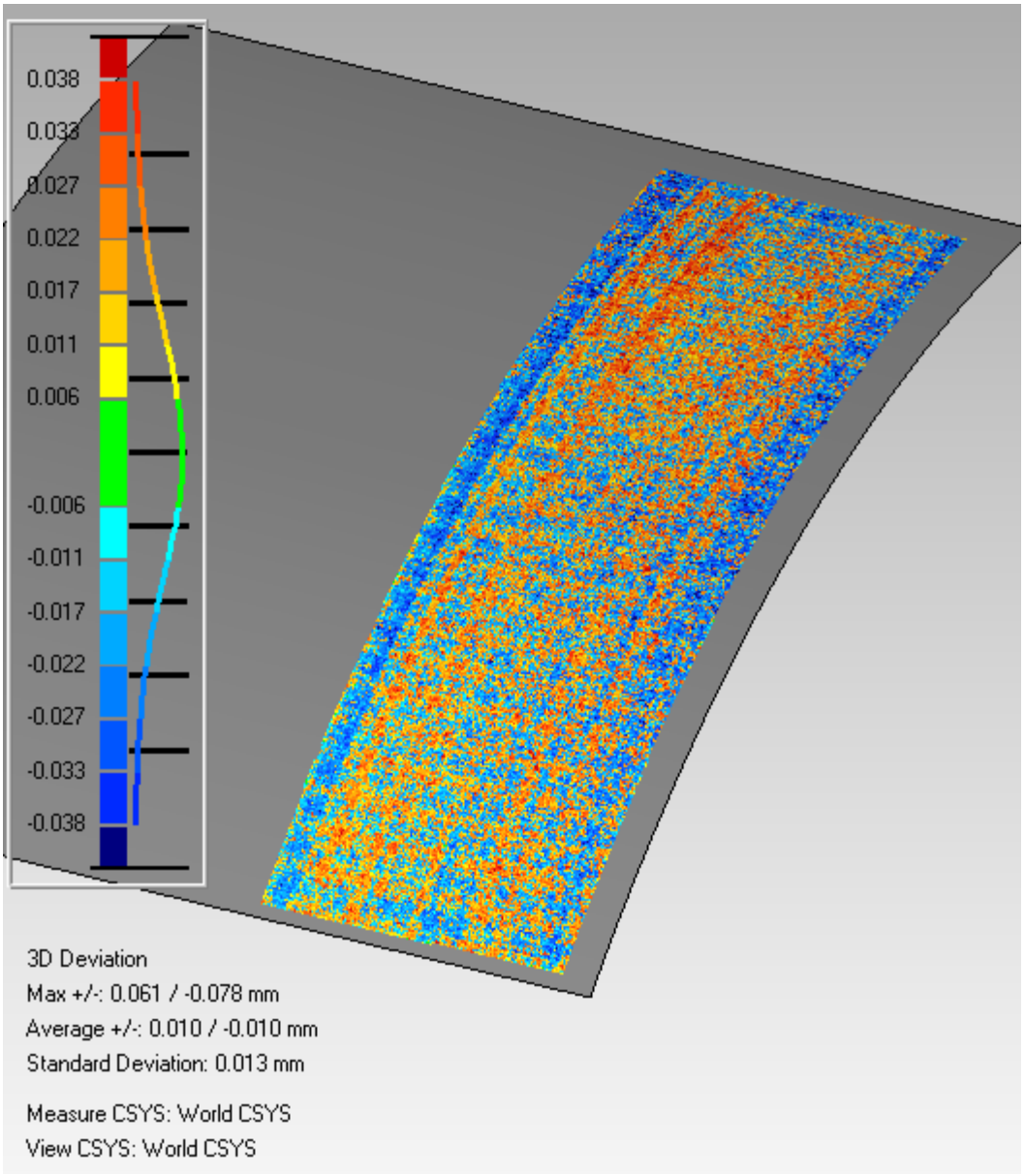


Figure 6.9 - Case 0, LMI Scan Data – Colour Map adjusted to show detail.

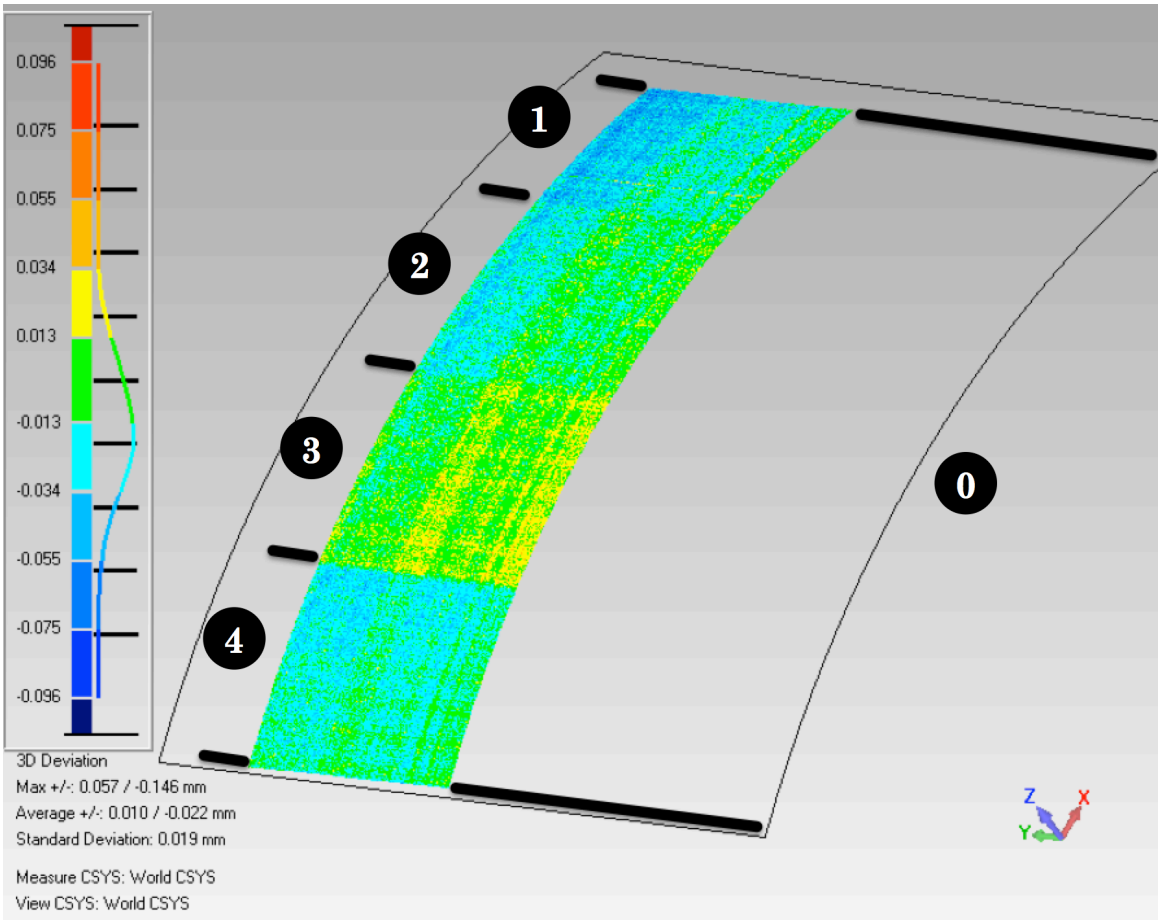


Figure 6.10 - LMI Scan Data: Cases 1-4 Surface Deviation Analysis

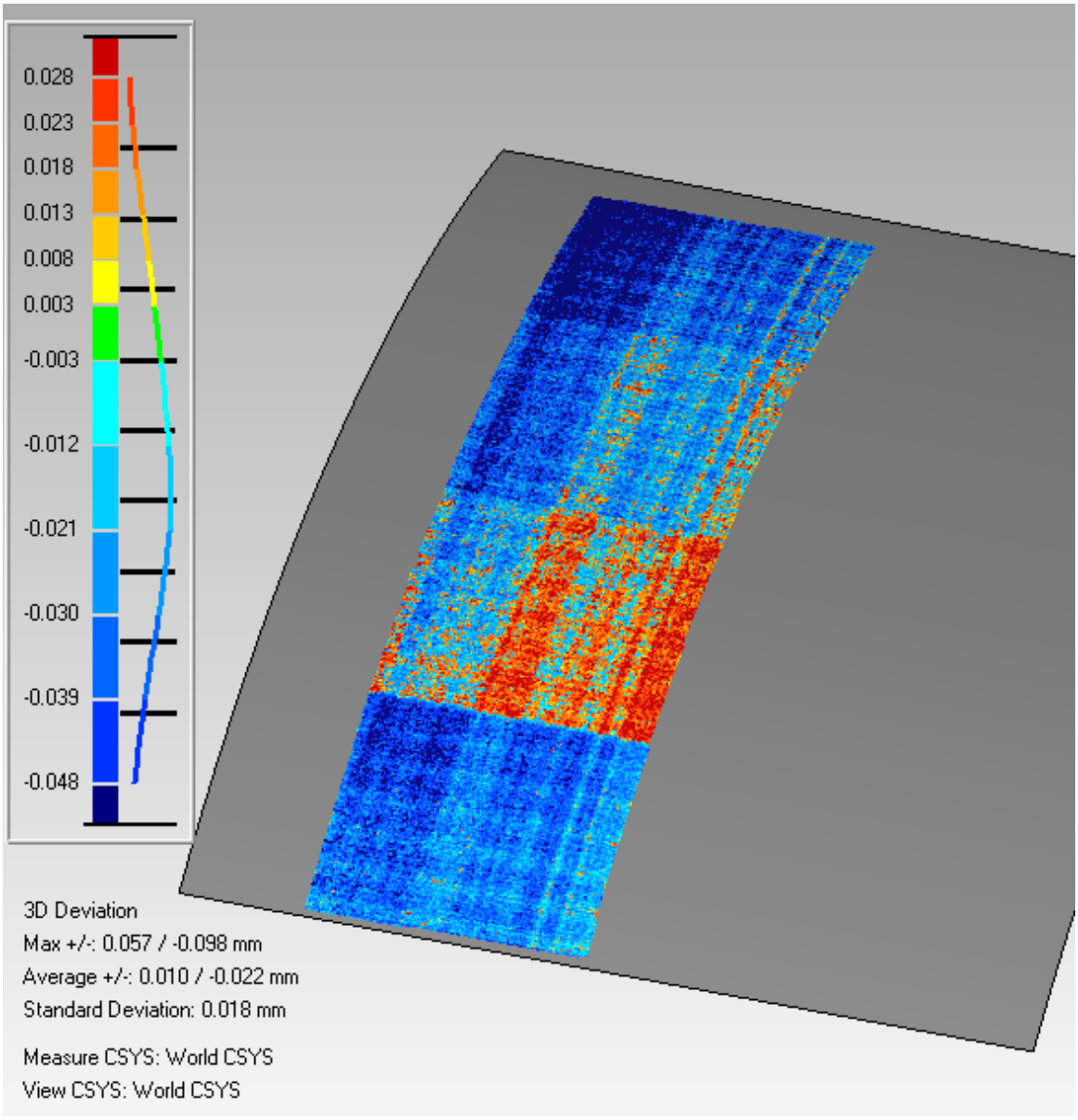


Figure 6.11 – Cases 1-4, LMI Scan Data – Colour Map adjusted to show detail.

Table 6.3 - LMI Scan Data Surface Fit Statistics

Average	+0.010mm	-0.022mm
Standard Deviation	0.019 mm	

The reference surface fit in Figure 6.8 showed a standard deviation of 13 μm. The manufacturer claimed Z resolution of the LMI scanner (0.006-0.014 mm) shows a much higher level of detail than the Roland scanner which claims an accuracy of 0.05mm. The higher Z resolution allows the final surface waviness characteristic of the ball nose end mill to be seen from the surface deviation report.

The scan density in this experiment is higher than required for successful fitting, but demonstrates the ability of laser scanning sensors to capture a point cloud much faster than is possible with a touch trigger probe measuring single points.

Chapter 7

Conclusions and Future Work

CNC machining has evolved from manually entered and tape punched instructions, to modern digitally stored and distributed programs. These systems currently operate on one-way data flow, and assume geometric consistency. In reality, factors such as casting variation, heat treatment distortion, and assembly creep wear contribute to workpiece geometry variations that require measurement required prior to machining. Scrap avoidance can be a reality by using CNC mounted inspection sensors, such as a touch trigger probe, together with existing inspection software employed with CMM inspection data. A method has been developed and demonstrated to allows a workpiece loaded in a CNC machine to be measured with a touch trigger probe, analyzed using GD&T software, and machined using a tool path updated based on the measurement and analysis.

In Chapter 4 the demonstrated pre-machining inspection procedure determined workpiece orientation in order to detect workpiece misalignment. A computer connected to the machining centre via a Universal Machine interface generated an updated toolpath for correct machining of the misaligned part. This process was tested using a fixturing plate with three spherical datums, fixtured in 4 different instances, and laser scan surface

analysis has shown an average deviation of +0.010/-0.022mm from nominal geometry, and a standard deviation of 0.019mm.

In Chapter 6 the feasibility of integrating a laser scanner with CNC was investigated using a CMM with optical linear encoders and a laser scanner with an Ethernet data interface. A computer application was created to link the captured 3D point data to the CMM location data, and a calibration sphere, and the workpiece from Chapter 4 were scanned with accuracy comparable to commercial scanning systems.

7.1 Future Work

The registration method in this work uses spherical datums, but there may be situations where this is not possible. Future work will include trials using part features as datums, and implementing the system on a 5-axis CNC machining centre.

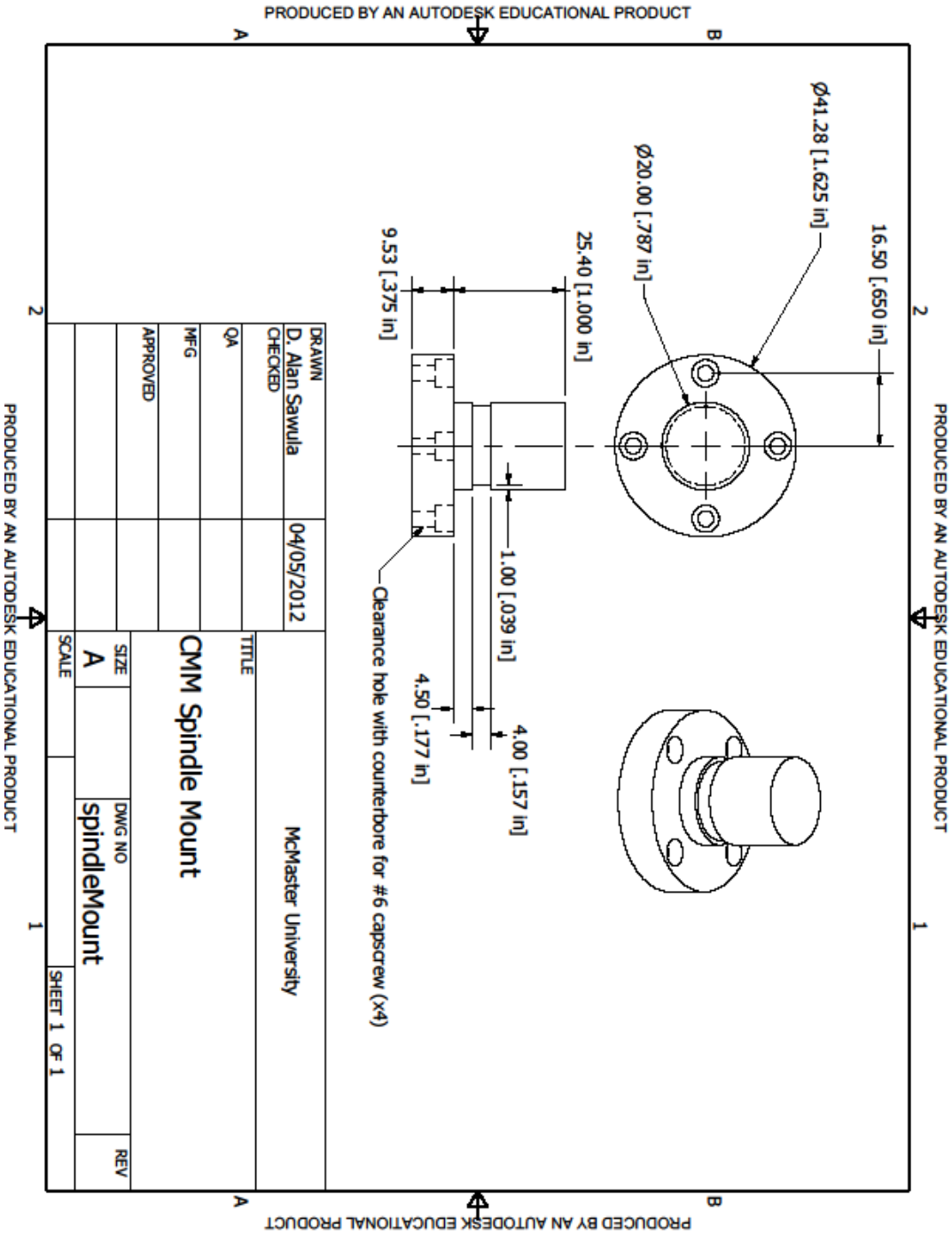
The use of a laser scanner in a CNC machine tool offers decreased inspection time due to higher data capture rates compared to contact-based digitizers. The position information required by the scan head may be obtained by connecting to the existing position feedback hardware, or by installing additional encoders that can operate in parallel. There are a number of challenges specific to optical inspection, such as reflectivity, the presence of coolant or chips, as well as liquid droplets on the lens of the

sensor. These may be overcome by methods used by other lasers in contaminating environments such as CNC grinding machines, where a compartment houses the scanner with a small opening allowing an un-restricted line of sight while air pressure keeps air-borne contaminants out.

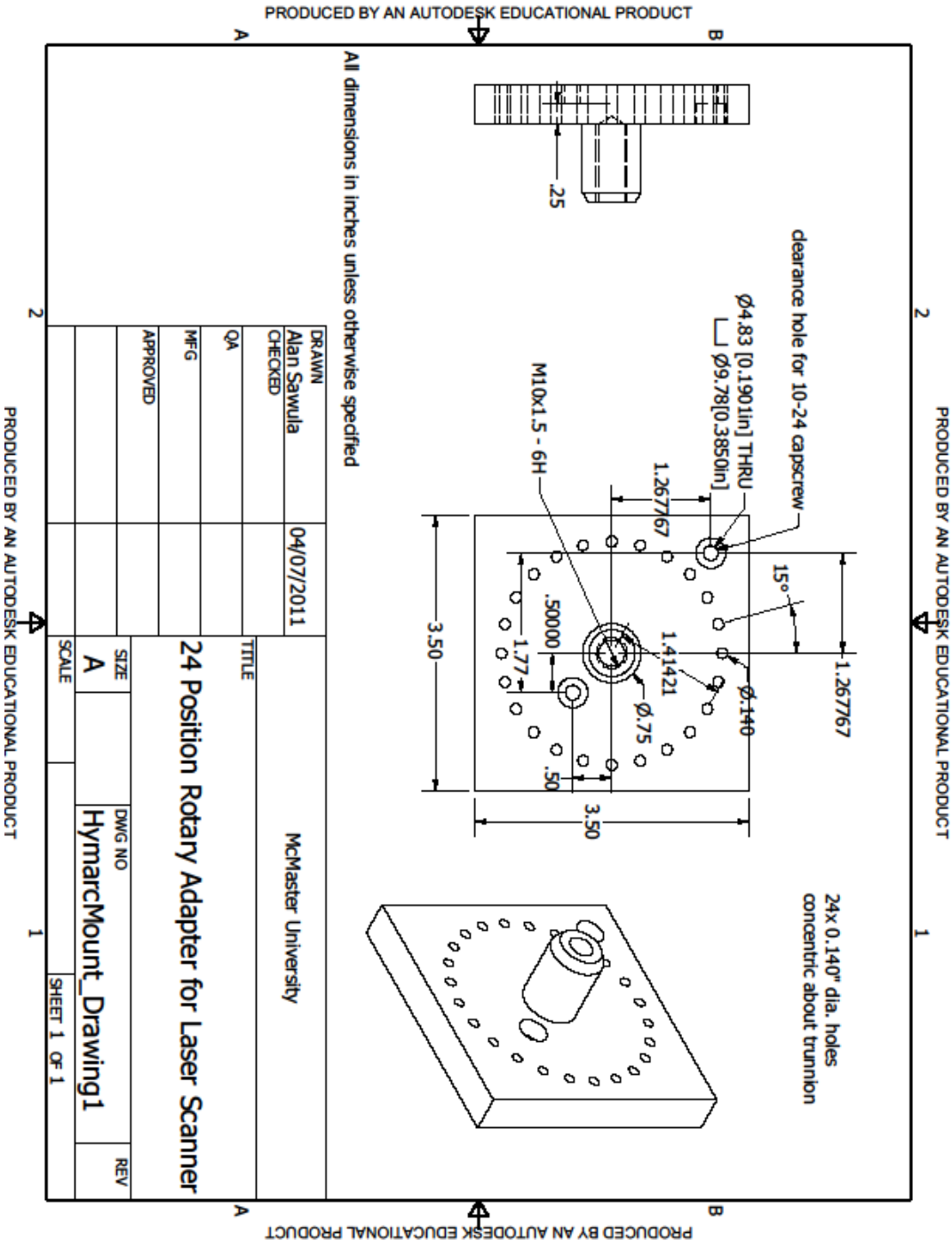
The most promising future work is the potential unlocked by having measurement and communication ability at the CNC control. The presence of advanced manufacturing software at the CNC control enables the design intentions of manufacturing engineers to be present as each part is machined.

Appendix

A. CMM Spindle Mount



B. 24 Position Rotary Adapter for CMM Mounted Laser Scanner



C. Ring Gauge Probing NC Code (First 50 Lines Only)

```
%
O1040
(A.SAWULA)
(RING GAUGE CALIBRATION)
G21 G40 G90 G54
M19
#31 = 100.0
POPEN
DPRNT[Index,X,Y]
PCLOS
G1 X0.0 Y0.0
(PROBE HIT1)
G31 X21.7889 Y249.0487
#1 = #5061
#2 = #5062
G1 X0.0 Y0.0
POPEN
DPRNT[1,#1 [44], #2 [44]
PCLOS
(PROBE HIT2)
G31 X43.4120 Y246.2019
#1 = #5061
#2 = #5062
G1 X0.0 Y0.0
POPEN
DPRNT[2,#1 [44], #2 [44]
PCLOS
(PROBE HIT3)
G31 X64.7048 Y241.4815
#1 = #5061
#2 = #5062
G1 X0.0 Y0.0
POPEN
DPRNT[3,#1 [44], #2 [44]
PCLOS
(PROBE HIT4)
G31 X85.5050 Y234.9232
#1 = #5061
#2 = #5062
G1 X0.0 Y0.0
POPEN
DPRNT[4,#1 [44], #2 [44]
PCLOS
(PROBE HIT5)
G31 X105.6546 Y226.5769
#1 = #5061
#2 = #5062
G1 X0.0 Y0.0
POPEN
DPRNT[5,#1 [44], #2 [44]
PCLOS
```

D. Sphere Probing Routine NC Code

```
%
O1050
(PROBES 1)
(MEASURE 1INCH DIA SPHERE)
(START AT XY NEAR CENTRE, Z ABOVE CENTRE)
(PROGRAM SETS X AND Y PROBE TRAJECTORIES BASED ON FIRST HIT IN Z-)
(CENTRE CALCULATED FROM THE AVERAGE OF PAIRS OF X AND Y HITS)
#20 = 5. (PROBE TIP DIA.)
#21 = 0.062 (PROBE FLEX COMPENS.0.062)
#22 = [#20/2]-#21
#23 = 273. (SPHERE SPACING)
#25 = 15.0 (SPHERE CLEARANCE)
#30 = 25.400 (SPHERE DIA.)
#31 = 100 (FAST PROBE SPEED)
#32 = 20 (SLOW PROBE SPEED)
#33 = 4000 (SAFE RAPID MOVE SPEED)
G21 G40 G90 G54
M19
G31 Z0.0 F#31 (PROBE Z)
#1 = #5061
#2 = #5062
#3 = [#5063-#20/2]
G1 Z[#3+#25] F#33 (UP)
G31 X[#1+[#30]+#20] F#33 (OVER+)
G31 Z[#3-[#30/2]] F#33 (DOWN)
G31 X[#1] F#31 (PROBE X1 FAST-)
G1 X[#5061+#22]F#33(RETRACT+)
G31 X[#1] F#32 (PROBE X1 SLOW-)
#4 = #5061-#22
#5 = #5062
#6 = #5063
G1 X[#4+#20] F#33 (RETRACT+)
G31 Z[#3+#25] F#33 (UP)
G31 X[#1-[#30]-#20] F#33 (OVER-)
G31 Z[#3-[#30/2]] F#33 (DOWN)
G31 X[#1] F#31 (PROBE X2 FAST+)
G1 X[#5061-#22](RETRACT+)
G31 X[#1] F#32 (PROBE X2 SLOW+)
#7 = #5061+#22
#8 = #5062
#9 = #5063
G1 X[#1-[#30]-#20] F#33 (RETRACT)
G31 Z[#3+#25] F#33 (UP)
G31 X[#1] Y[#2+[#30]+#20] F#33 (OVER+)
G31 Z[#3-[#30/2]] F#33 (DOWN)
G31 Y[#2] F#31 (PROBE Y1 FAST-)
G1 Y[#5062+#22](RETRACT+)
G31 Y[#2] F#32 (PROBE Y1 SLOW-)
#10 = #5061
#11 = #5062-#22
#12 = #5063
G1 Y[#2+[#30]+#20] F#33 (RETRACT+)
```

```
G31 Z[#3+#25] F#33 (UP)
G31 Y[#2-[#30]-#20] F#33 (OVER-)
G31 Z[#3-[#30/2]] F#33 (DOWN)
G31 Y[#2] F#31 (PROBE Y2 FAST+)
G1 Y[#5062-#22](RETRACT-)
G31 Y[#2] F#32 (PROBE Y1 SLOW+)
#13 = #5061
#14 = #5062+#22
#15 = #5063
G1 Y[#2-[#30]-#20] F#33 (RETRACT-)
G31 Z[#3+#25] F#33 (UP)
#16=[#4+#7]/2 (X)
#17=[#11+#14]/2 (Y)
#18= #4-#7
(TO CENTRE)
G31 X#16 Y#17 F#33
G31 Z[#3-150.0] F#31
#1= #5061
#2= #5062
#3 = #5063-#20/2
#19 = #3-[#30/2]
G1 Z[#5063+#20]  F#33
G31 Z[#3-150.0] F#31 (PROBE Z)
#1 = #5061
#2 = #5062
#3 = [#5063-#20/2]
G1 Z[#3+#25] F#33 (UP)
G31 X[#1+[#30]+#20] F#33 (OVER+)
G31 Z[#3-[#30/2]] F#33 (DOWN)
G31 X[#1] F#31 (PROBE X1 FAST-)
G1 X[#5061+#22](RETRACT+)
G31 X[#1] F#32 (PROBE X1 SLOW-)
#4 = #5061-#22
#5 = #5062
#6 = #5063
G1 X[#4+#20] F#33 (RETRACT+)
G31 Z[#3+#25] F#33 (UP)
G31 X[#1-[#30]-#20] F#33 (OVER-)
G31 Z[#3-[#30/2]] F#33 (DOWN)
G31 X[#1] F#31 (PROBE X2 FAST+)
G1 X[#5061-#22](RETRACT+)
G31 X[#1] F#32 (PROBE X2 SLOW+)
#7 = #5061+#22
#8 = #5062
#9 = #5063
G1 X[#1-[#30]-#20] F#33 (RETRACT)
G31 Z[#3+#25] F#33 (UP)
G31 X[#1] Y[#2+[#30]+#20] F#33 (OVER+)
G31 Z[#3-[#30/2]] F#33 (DOWN)
G31 Y[#2] F#31 (PROBE Y1 FAST-)
G1 Y[#5062+#22](RETRACT+)
G31 Y[#2] F#32 (PROBE Y1 SLOW-)
#10 = #5061
#11 = #5062-#22
#12 = #5063
```

```
G1 Y[#2+[#30]+#20] F#33 (RETRACT+)
G31 Z[#3+#25] F#33 (UP)
G31 Y[#2-[#30]-#20] F#33 (OVER-)
G31 Z[#3-[#30/2]] F#33 (DOWN)
G31 Y[#2] F#31 (PROBE Y2 FAST+)
G1 Y[#5062-#22](RETRACT-)
G31 Y[#2] F#32 (PROBE Y2 SLOW+)
#13 = #5061
#14 = #5062+#22
#15 = #5063
G1 Y[#2-[#30]-#20] F#33 (RETRACT-)
G31 Z[#3+#25] F#33 (UP)
#16=[#4+#7]/2 (X)
#17=[#11+#14]/2 (Y)
#18= #4-#7
(TO CENTRE)
G31 X#16 Y#17 F#33
G31 Z[#3-150.0] F#31
#1= #5061
#2= #5062
#3 = #5063-[#20/2]
#19 = #3-[#30/2]
G1 Z[#5063+#20] F#33
POPEN
DPRNT[SPHERE1]
DPRNT[***]
DPRNT[5]
DPRNT[*#1 [44]* #2 [44]* #3 [44]]
DPRNT[*#4 [44]* #5 [44]* #6 [44]]
DPRNT[*#7 [44]* #8 [44]* #9 [44]]
DPRNT[*#10 [44]* #11 [44]* #12 [44]]
DPRNT[*#13 [44]* #14 [44]* #15 [44]]
DPRNT[***]
PCLOS
M30
%
```

E. Nominal Tool Path from CAM Software (first 50 Lines only)

```
%
:0001
(BladeBaseMtl2012ZUp)
N10 G21 G90 G40
N20 G10 P1 Z85.0 R4.0 T00
N30 G10 L2 P1 X0.0 Y0.0 Z0.0 (Top)
N40 (DEFINE OPERATION : PARALLEL LACE OPERATION)
N50 G0 X0.0 Y0.0 Z100.0
N60 M5
N70 G28 G91 Z0 H0
N80 G28 X0 Y0
O90 G21 G90 G40 G94
N100 T00 M06 (8.0 MM DIA BALL NOSE MILL)
N110 G54P1
N120 T00 M1
N130 S5000 M3 M42
N140 G0 X0.0 Y0.0
N150 G43 Z50.0 H00 M7
N160 G0 X-25.399 Y35.004 Z50.0
N170 G0 X-25.399 Y35.004 Z23.486
N180 G1 X-25.399 Y35.004 Z18.486 F1000.0
N190 G1 X25.391 Y35.004 Z18.486
N200 G0 X25.391 Y35.004 Z50.0
N210 G0 X-25.399 Y34.48 Z50.0
N220 G0 X-25.399 Y34.48 Z23.698
N230 G1 X-25.399 Y34.48 Z18.698
N240 G1 X25.391 Y34.48 Z18.698
N250 G0 X25.391 Y34.48 Z50.0
N260 G0 X-25.399 Y33.955 Z50.0
N270 G0 X-25.399 Y33.955 Z23.909
N280 G1 X-25.399 Y33.955 Z18.909
N290 G1 X25.391 Y33.955 Z18.909
N300 G0 X25.391 Y33.955 Z50.0
N310 G0 X-25.399 Y33.431 Z50.0
N320 G0 X-25.399 Y33.431 Z24.12
N330 G1 X-25.399 Y33.431 Z19.12
N340 G1 X25.391 Y33.431 Z19.12
N350 G0 X25.391 Y33.431 Z50.0
N360 G0 X-25.399 Y32.901 Z50.0
N370 G0 X-25.399 Y32.901 Z24.316
N380 G1 X-25.399 Y32.901 Z19.316
N390 G1 X25.391 Y32.901 Z19.316
N400 G0 X25.391 Y32.901 Z50.0
N410 G0 X-25.399 Y32.37 Z50.0
N420 G0 X-25.399 Y32.37 Z24.511
N430 G1 X-25.399 Y32.37 Z19.511
N440 G1 X25.391 Y32.37 Z19.511
N450 G0 X25.391 Y32.37 Z50.0
N460 G0 X-25.399 Y31.84 Z50.0
N470 G0 X-25.399 Y31.84 Z24.707
```

F. Transformed Tool Path (First 50 Lines only)

```
%
O1064
(ALAN-S-HUB4)
G21 G90 G40
(DEFINE OPERATION : PARALLEL LACE OPERATION)
G0 X24.886 Y0.215 Z35.016
G21 G90 G40 G94 (8.0 MM DIA BALL NOSE MILL)
G54
S5000 M3
G0 X24.886 Y0.215 Z35.016
G0 X10.972 Y25.275 Z-14.973
G0 X10.966 Y25.281 Z-35.588
G1 X10.964 Y25.283 Z-40.588 F250.0
G1 X12.282 Y-25.493 Z-40.603
G0 X12.290 Y-25.500 Z-14.988
G0 X10.412 Y25.261 Z-14.973
G0 X10.406 Y25.267 Z-35.671
G1 X10.404 Y25.268 Z-40.671
G1 X11.722 Y-25.508 Z-40.685
G0 X11.730 Y-25.515 Z-14.987
G0 X9.853 Y25.246 Z-14.972
G0 X9.847 Y25.252 Z-35.753
G1 X9.846 Y25.254 Z-40.753
G1 X11.163 Y-25.522 Z-40.768
G0 X11.171 Y-25.529 Z-14.987
G0 X9.294 Y25.232 Z-14.972
G0 X9.288 Y25.238 Z-35.836
G1 X9.287 Y25.239 Z-40.836
G1 X10.605 Y-25.537 Z-40.851
G0 X10.612 Y-25.544 Z-14.987
G0 X8.738 Y25.217 Z-14.972
G0 X8.731 Y25.223 Z-35.933
G1 X8.730 Y25.225 Z-40.933
G1 X10.048 Y-25.551 Z-40.948
G0 X10.056 Y-25.558 Z-14.987
G0 X8.181 Y25.203 Z-14.972
G0 X8.175 Y25.209 Z-36.033
G1 X8.173 Y25.210 Z-41.033
G1 X9.491 Y-25.565 Z-41.048
G0 X9.499 Y-25.573 Z-14.987
G0 X7.625 Y25.189 Z-14.972
G0 X7.619 Y25.195 Z-36.132
G1 X7.617 Y25.196 Z-41.132
G1 X8.935 Y-25.580 Z-41.147
G0 X8.943 Y-25.587 Z-14.987
G0 X7.068 Y25.174 Z-14.972
G0 X7.062 Y25.180 Z-36.231
G1 X7.060 Y25.182 Z-41.231
G1 X8.378 Y-25.594 Z-41.246
G0 X8.386 Y-25.602 Z-14.987
G0 X6.512 Y25.160 Z-14.97
```

G. CMM Laser Scanner Application Source Code

```
1 #pragma once
2
3
4 #include "stdafx.h"
5 #include "Form1.h"
6 #include <iostream>
7 #include <fstream>
8 using namespace std;
9
10 //cmm motion communication
11 #include <winsock2.h>
12 #include <ws2tcpip.h>
13 #include <stdio.h>
14
15 //encoder quadrature capture card
16 #include <conio.h>
17 #include "..\cbw.h"
18
19 //scanner
20 #include <Go2.h>
21 #include <stdlib.h>
22 #include <memory.h>
23 #include <windows.h>
24 #include <time.h>
25
26 #pragma comment(lib, "Ws2_32.lib")
27 #pragma comment(lib, "comdlg32.lib")
28 #pragma comment(lib, "Go2.lib")
29 #pragma comment(lib, "user32.lib")
30
31 #define RECEIVE_TIMEOUT    20000
32 #define INVALID_RANGE_16BIT      (0x8000)    //   gocator   transmits
range data as 16-bit signed integers. 0x8000 signifies invalid range
data.
33 #define DOUBLE_MAX   (1.79769e+308)  //  64-bit  double  -  largest
positive value.
34 #define INVALID_RANGE_DOUBLE     (-DOUBLE_MAX)     // floating point
value to represent invalid range data.
35
36 SOCKET ConnectSocket = INVALID_SOCKET;
37 int iResult;
38 int testCount = 0;
39 /* Encoder Variable Declarations */
40 //int Row, Col;
41 int ULStat = 0;
42 int BoardNum = 0;
43 int CounterNum, Quadrature, CountingMode, DataEncoding, IndexMode;
44 int InvertIndex, FlagPins, GateEnable;
45 long LoadValue;
46 unsigned long Count, lastCount, StatusBits;
47 unsigned long xCount, xlastCount, xStatusBits;
```

```
48 unsigned long yCount, ylastCount, yStatusBits;
49 unsigned long zCount, zlastCount, zStatusBits;
50 int RegName;
51 float RevLevel = (float)CURRENTREVNUM;
52 char *DirectionStr;
53 //Scanner Variable Declarations
54 typedef struct
55 {
56    short x;    // x-coordinate as 16-bit signed integer - position
along laser line
57    short z;    // z-coordinate as 16-bit signed integer - height (at
the given x position)
58 }ProfilePoint16s;
59
60 typedef struct
61 {
62    double x;   // x-coordinate in engineering units (mm) - position
along laser line
63    double y;   // y-coordinate in engineering units (mm) - derived
from scanner location
64    double z;   // z-coordinate in engineering units (mm) - height (at
the given x position)
65 }ProfilePoint;
66 int status;
67 Go2System scannerSystem = GO2_NULL;
68 Go2Data data = GO2_NULL;
69 Go2IPAddress address;
70 ProfilePoint ** memory = NULL;
71 int j;
72 int i;
73 unsigned int arrayIndex;
74 unsigned int timeout = 20000;
75 double XScannerOffset = 0;
76 double YScannerOffset = 0;
77 double ZScannerOffset = 0;
78 double YMax = 400;
79 double ZMax = 230;
80
81 int scanCount = 200* 120;//200 hz * scanDurationSeconds
82 int scanIndex = 0; //stores the location where to store the next scan
data
83 clock_t start, end;
84 double cpu_time;
85
86
87
88 namespace SimpleForm {
89
90    using namespace System;
91    using namespace System::ComponentModel;
92    using namespace System::Collections;
93    using namespace System::Windows::Forms;
94    using namespace System::Data;
95    using namespace System::Drawing;
96    using namespace System::Runtime::InteropServices;
```

```
97
98    /// <summary>
99    /// Summary for Form1
100   /// </summary>
101   public ref class Form1 : public System::Windows::Forms::Form
102   {
103   public:
104   Form1(void)
105   {
106   InitializeComponent();
107
108   }
109
110   protected:
111   ~Form1()
112   {
113   // shutdown the connection for sending since no more data will be
sent
114   // the client can still use the ConnectSocket for receiving data
115   iResult = shutdown(ConnectSocket, SD_SEND);
116   if (iResult == SOCKET_ERROR) {
117   printf("shutdown failed: %d\n", WSAGetLastError());
118   closesocket(ConnectSocket);
119   WSACleanup();
120   // return 1;
121   }
122
123   //Scanner Cleanup
124   // stop Gocator system
125   if ((status = Go2System_Stop(scannerSystem)) != GO2_OK)
126   {
127   printf ("Error: Go2System_Stop:%d\n", status);
128   return;
129   }
130
131   // destroy system handle
132   if ((status = Go2System_Destroy(scannerSystem)) != GO2_OK)
133   {
134   printf ("Error: Go2System_Destroy:%d\n", status);
135   return;
136   }
137
138   // terminate Go2 API
139   if ((status = Go2Api_Terminate()) != GO2_OK)
140   {
141   printf ("Error: Go2Api_Terminate:%d\n", status);
142   return;
143   }
144   printf("Press any key to Free Memory...\n", status);
145   getchar();
146   //free memory array
147   for(i = 0; i < scanCount; i++)
148   free(memory[i]);
149   free(memory);
150   //End Scanner Cleanup
```

```
151
152   if (components)
153   {
154   delete components;
155   }
156   }
157   private: System::Windows::Forms::Button^ button1;
158   protected:
159
160
161
162   public: System::Windows::Forms::TextBox^ textBox1;
163   public: System::Windows::Forms::Label^ label2;
164   public: System::Windows::Forms::RichTextBox^ richTextBox1;
165   public: System::Windows::Forms::Label^ staticStatusLabel;
166   public: System::Windows::Forms::Label^ statusLabel;
167   public: System::Windows::Forms::Button^ button2;
168
169   public: System::Windows::Forms::Label^ label3;
170   public: System::Windows::Forms::Label^ label4;
171   public: System::Windows::Forms::Label^ label5;
172
173   public: System::Windows::Forms::Label^ xlabel;
174   public: System::Windows::Forms::Label^ ylabel;
175   public: System::Windows::Forms::Label^ zlabel;
176   private: System::Windows::Forms::GroupBox^ groupBox1;
177   public: System::Windows::Forms::Timer^ timer1;
178   public: System::Windows::Forms::CheckBox^ checkBox1;
179   public: System::Windows::Forms::RichTextBox^ richTextBox2;
180   public: System::Windows::Forms::Button^ buttonStartScan;
181   public: System::Windows::Forms::Button^ buttonStopScan;
182   private: System::Windows::Forms::Button^ buttonSaveData;
183   private: System::Windows::Forms::Timer^ timerScan;
184   public: System::Windows::Forms::Label^ labelScanSize;
185   private: System::Windows::Forms::Label^ label1;
186   public: System::Windows::Forms::ProgressBar^ progressBar1;
187   private: System::Windows::Forms::TrackBar^ trackBar1;
188   public: System::Windows::Forms::Label^ labelScanRateHz;
189   private: System::Windows::Forms::Button^ buttonRestartScan;
190   protected:
191   private: System::ComponentModel::IContainer^ components;
192
193
194 #pragma region Windows Form Designer generated code
195   /// <summary>
196   /// Required method for Designer support - do not modify
197   /// the contents of this method with the code editor.
198   /// </summary>
199   void InitializeComponent(void)
200   {
201   this->components = (gcnew System::ComponentModel::Container());
202   this->button1 = (gcnew System::Windows::Forms::Button());
203   this->textBox1 = (gcnew System::Windows::Forms::TextBox());
204   this->label2 = (gcnew System::Windows::Forms::Label());
```

```
205   this->richTextBox1                    =                    (gcnew
System::Windows::Forms::RichTextBox());
206   this->staticStatusLabel = (gcnew System::Windows::Forms::Label());
207   this->statusLabel = (gcnew System::Windows::Forms::Label());
208   this->button2 = (gcnew System::Windows::Forms::Button());
209   this->label3 = (gcnew System::Windows::Forms::Label());
210   this->label4 = (gcnew System::Windows::Forms::Label());
211   this->label5 = (gcnew System::Windows::Forms::Label());
212   this->xlabel = (gcnew System::Windows::Forms::Label());
213   this->ylabel = (gcnew System::Windows::Forms::Label());
214   this->zlabel = (gcnew System::Windows::Forms::Label());
215   this->groupBox1 = (gcnew System::Windows::Forms::GroupBox());
216   this->checkBox1 = (gcnew System::Windows::Forms::CheckBox());
217   this->timer1    =    (gcnew    System::Windows::Forms::Timer(this-
>components));
218   this->richTextBox2                    =                    (gcnew
System::Windows::Forms::RichTextBox());
219   this->buttonStartScan = (gcnew System::Windows::Forms::Button());
220   this->buttonStopScan = (gcnew System::Windows::Forms::Button());
221   this->buttonSaveData = (gcnew System::Windows::Forms::Button());
222   this->timerScan   =   (gcnew   System::Windows::Forms::Timer(this-
>components));
223   this->labelScanSize = (gcnew System::Windows::Forms::Label());
224   this->label1 = (gcnew System::Windows::Forms::Label());
225   this->progressBar1                    =                    (gcnew
System::Windows::Forms::ProgressBar());
226   this->trackBar1 = (gcnew System::Windows::Forms::TrackBar());
227   this->labelScanRateHz = (gcnew System::Windows::Forms::Label());
228   this->buttonRestartScan                 =                    (gcnew
System::Windows::Forms::Button());
229   this->groupBox1->SuspendLayout();
230   (cli::safe_cast<System::ComponentModel::ISupportInitialize^
>(this->trackBar1))->BeginInit();
231   this->SuspendLayout();
232   //
233   // button1
234   //
235   this->button1->Location = System::Drawing::Point(288, 268);
236   this->button1->Name = L"button1";
237   this->button1->Size = System::Drawing::Size(75, 23);
238   this->button1->TabIndex = 0;
239   this->button1->Text = L"Send";
240   this->button1->UseVisualStyleBackColor = true;
241   this->button1->Click    +=    gcnew    System::EventHandler(this,
&Form1::button1_Click);
242   //
243   // textBox1
244   //
245   this->textBox1->AcceptsReturn = true;
246   this->textBox1->HideSelection = false;
247   this->textBox1->Location = System::Drawing::Point(69, 270);
248   this->textBox1->Name = L"textBox1";
249   this->textBox1->Size = System::Drawing::Size(213, 20);
250   this->textBox1->TabIndex = 2;
251   this->textBox1->Text = L"HECK2";
```

```
252   this->textBox1->TextChanged  +=  gcnew  System::EventHandler(this,
&Form1::textBox1_TextChanged);
253   //
254   // label2
255   //
256   this->label2->AutoSize = true;
257   this->label2->Location = System::Drawing::Point(9, 273);
258   this->label2->Name = L"label2";
259   this->label2->Size = System::Drawing::Size(57, 13);
260   this->label2->TabIndex = 3;
261   this->label2->Text = L"Command:";
262   //
263   // richTextBox1
264   //
265   this->richTextBox1->Location = System::Drawing::Point(9, 34);
266   this->richTextBox1->Name = L"richTextBox1";
267   this->richTextBox1->Size = System::Drawing::Size(271, 227);
268   this->richTextBox1->TabIndex = 5;
269   this->richTextBox1->Text = L"";
270   //
271   // staticStatusLabel
272   //
273   this->staticStatusLabel->AutoSize = true;
274   this->staticStatusLabel->Location = System::Drawing::Point(8, 9);
275   this->staticStatusLabel->Name = L"staticStatusLabel";
276   this->staticStatusLabel->Size = System::Drawing::Size(40, 13);
277   this->staticStatusLabel->TabIndex = 6;
278   this->staticStatusLabel->Text = L"Status:";
279   this->staticStatusLabel->Click += gcnew System::EventHandler(this,
&Form1::label3_Click);
280   //
281   // statusLabel
282   //
283   this->statusLabel->AutoSize = true;
284   this->statusLabel->Location = System::Drawing::Point(54, 9);
285   this->statusLabel->Name = L"statusLabel";
286   this->statusLabel->Size = System::Drawing::Size(228, 13);
287   this->statusLabel->TabIndex = 7;
288   this->statusLabel->Text  =  L"Unable  to  connect  -  Please  Restart
Application";
289   //
290   // button2
291   //
292   this->button2->AutoSize = true;
293   this->button2->Location = System::Drawing::Point(20, 57);
294   this->button2->Name = L"button2";
295   this->button2->Size = System::Drawing::Size(118, 28);
296   this->button2->TabIndex = 8;
297   this->button2->Text = L"Set Position as Home";
298   this->button2->UseVisualStyleBackColor = true;
299   this->button2->Click    +=    gcnew    System::EventHandler(this,
&Form1::button2_Click);
300   //
301   // label3
302   //
```

```
303   this->label3->AutoSize = true;
304   this->label3->Font   =   (gcnew   System::Drawing::Font(L"Calibri",
27.75F, System::Drawing::FontStyle
::Bold));
305   this->label3->Location = System::Drawing::Point(18, 94);
306   this->label3->Name = L"label3";
307   this->label3->Size = System::Drawing::Size(50, 45);
308   this->label3->TabIndex = 9;
309   this->label3->Text = L"X:";
310   //
311   // label4
312   //
313   this->label4->AutoSize = true;
314   this->label4->Font   =   (gcnew   System::Drawing::Font(L"Calibri",
27.75F, System::Drawing::FontStyle
::Bold));
315   this->label4->Location = System::Drawing::Point(18, 151);
316   this->label4->Name = L"label4";
317   this->label4->Size = System::Drawing::Size(46, 45);
318   this->label4->TabIndex = 10;
319   this->label4->Text = L"Y:";
320   //
321   // label5
322   //
323   this->label5->AutoSize = true;
324   this->label5->Font   =   (gcnew   System::Drawing::Font(L"Calibri",
27.75F, System::Drawing::FontStyle
::Bold, System::Drawing::GraphicsUnit::Point,
325   static_cast<System::Byte>(0)));
326   this->label5->Location = System::Drawing::Point(18, 207);
327   this->label5->Name = L"label5";
328   this->label5->Size = System::Drawing::Size(48, 45);
329   this->label5->TabIndex = 11;
330   this->label5->Text = L"Z:";
331   //
332   // xlabel
333   //
334   this->xlabel->AutoSize = true;
335   this->xlabel->Font   =   (gcnew   System::Drawing::Font(L"Calibri",
27.75F, System::Drawing::FontStyle
::Bold));
336   this->xlabel->Location = System::Drawing::Point(74, 94);
337   this->xlabel->Name = L"xlabel";
338   this->xlabel->Size = System::Drawing::Size(134, 45);
339   this->xlabel->TabIndex = 12;
340   this->xlabel->Text = L"123456";
341   //
342   // ylabel
343   //
344   this->ylabel->AutoSize = true;
345   this->ylabel->Font   =   (gcnew   System::Drawing::Font(L"Calibri",
27.75F, System::Drawing::FontStyle
::Bold));
346   this->ylabel->Location = System::Drawing::Point(74, 151);
347   this->ylabel->Name = L"ylabel";
```

```
348   this->ylabel->Size = System::Drawing::Size(134, 45);
349   this->ylabel->TabIndex = 13;
350   this->ylabel->Text = L"123456";
351   //
352   // zlabel
353   //
354   this->zlabel->AutoSize = true;
355   this->zlabel->Font   =   (gcnew   System::Drawing::Font(L"Calibri",
27.75F, System::Drawing::FontStyle
::Bold));
356   this->zlabel->Location = System::Drawing::Point(74, 207);
357   this->zlabel->Name = L"zlabel";
358   this->zlabel->Size = System::Drawing::Size(134, 45);
359   this->zlabel->TabIndex = 14;
360   this->zlabel->Text = L"123456";
361   //
362   // groupBox1
363   //
364   this->groupBox1->Controls->Add(this->xlabel);
365   this->groupBox1->Controls->Add(this->checkBox1);
366   this->groupBox1->Controls->Add(this->zlabel);
367   this->groupBox1->Controls->Add(this->button2);
368   this->groupBox1->Controls->Add(this->label3);
369   this->groupBox1->Controls->Add(this->ylabel);
370   this->groupBox1->Controls->Add(this->label4);
371   this->groupBox1->Controls->Add(this->label5);
372   this->groupBox1->Location = System::Drawing::Point(413, 34);
373   this->groupBox1->Name = L"groupBox1";
374   this->groupBox1->Size = System::Drawing::Size(239, 441);
375   this->groupBox1->TabIndex = 15;
376   this->groupBox1->TabStop = false;
377   this->groupBox1->Text = L"CMM Coordinates";
378   //
379   // checkBox1
380   //
381   this->checkBox1->AutoSize = true;
382   this->checkBox1->Location = System::Drawing::Point(20, 34);
383   this->checkBox1->Name = L"checkBox1";
384   this->checkBox1->Size = System::Drawing::Size(101, 17);
385   this->checkBox1->TabIndex = 16;
386   this->checkBox1->Text = L"Monitor Position";
387   this->checkBox1->UseVisualStyleBackColor = true;
388   this->checkBox1->CheckedChanged              +=              gcnew
System::EventHandler(this, &Form1::
checkBox1_CheckedChanged);
389   //
390   // timer1
391   //
392   this->timer1->Interval = 5;
393   this->timer1->Tick     +=     gcnew     System::EventHandler(this,
&Form1::timer1_Tick);
394   //
395   // richTextBox2
396   //
397   this->richTextBox2->Location = System::Drawing::Point(286, 35);
```

```
398   this->richTextBox2->Name = L"richTextBox2";
399   this->richTextBox2->Size = System::Drawing::Size(121, 227);
400   this->richTextBox2->TabIndex = 17;
401   this->richTextBox2->Text = L"";
402   //
403   // buttonStartScan
404   //
405   this->buttonStartScan->Location = System::Drawing::Point(42, 332);
406   this->buttonStartScan->Name = L"buttonStartScan";
407   this->buttonStartScan->Size = System::Drawing::Size(123, 37);
408   this->buttonStartScan->TabIndex = 18;
409   this->buttonStartScan->Text = L"Start Scan";
410   this->buttonStartScan->UseVisualStyleBackColor = true;
411   this->buttonStartScan->Click  +=  gcnew  System::EventHandler(this,
&Form1::buttonStartScan_Click);
412   //
413   // buttonStopScan
414   //
415   this->buttonStopScan->Enabled = false;
416   this->buttonStopScan->Location = System::Drawing::Point(42, 375);
417   this->buttonStopScan->Name = L"buttonStopScan";
418   this->buttonStopScan->Size = System::Drawing::Size(123, 37);
419   this->buttonStopScan->TabIndex = 19;
420   this->buttonStopScan->Text = L"Stop Scan";
421   this->buttonStopScan->UseVisualStyleBackColor = true;
422   this->buttonStopScan->Click  +=  gcnew  System::EventHandler(this,
&Form1::buttonStopScan_Click);
423   //
424   // buttonSaveData
425   //
426   this->buttonSaveData->Location = System::Drawing::Point(226, 437);
427   this->buttonSaveData->Name = L"buttonSaveData";
428   this->buttonSaveData->Size = System::Drawing::Size(123, 38);
429   this->buttonSaveData->TabIndex = 20;
430   this->buttonSaveData->Text = L"Save Data";
431   this->buttonSaveData->UseVisualStyleBackColor = true;
432   this->buttonSaveData->Click  +=  gcnew  System::EventHandler(this,
&Form1::buttonSaveData_Click);
433   //
434   // timerScan
435   //
436   this->timerScan->Interval = 5;
437   this->timerScan->Tick    +=    gcnew    System::EventHandler(this,
&Form1::timerScan_Tick);
438   //
439   // labelScanSize
440   //
441   this->labelScanSize->AutoSize = true;
442   this->labelScanSize->Location = System::Drawing::Point(323, 375);
443   this->labelScanSize->Name = L"labelScanSize";
444   this->labelScanSize->Size = System::Drawing::Size(59, 13);
445   this->labelScanSize->TabIndex = 21;
446   this->labelScanSize->Text = L"ScanSize#";
447   //
448   // label1
```

```
449   //
450   this->label1->AutoSize = true;
451   this->label1->Location = System::Drawing::Point(223, 375);
452   this->label1->Name = L"label1";
453   this->label1->Size = System::Drawing::Size(98, 13);
454   this->label1->TabIndex = 22;
455   this->label1->Text = L"Scan Profile Count:";
456   //
457   // progressBar1
458   //
459   this->progressBar1->Location = System::Drawing::Point(226, 408);
460   this->progressBar1->MarqueeAnimationSpeed = 1000;
461   this->progressBar1->Name = L"progressBar1";
462   this->progressBar1->Size = System::Drawing::Size(123, 23);
463   this->progressBar1->Step = 1;
464   this->progressBar1->Style                                          =
System::Windows::Forms::ProgressBarStyle::Continuous;
465   this->progressBar1->TabIndex = 23;
466   //
467   // trackBar1
468   //
469   this->trackBar1->Location = System::Drawing::Point(202, 327);
470   this->trackBar1->Maximum = 200;
471   this->trackBar1->Name = L"trackBar1";
472   this->trackBar1->Size = System::Drawing::Size(205, 45);
473   this->trackBar1->TabIndex = 24;
474   this->trackBar1->Value = 200;
475   this->trackBar1->Scroll   +=   gcnew   System::EventHandler(this,
&Form1::trackBar1_Scroll);
476   //
477   // labelScanRateHz
478   //
479   this->labelScanRateHz->AutoSize = true;
480   this->labelScanRateHz->Location   =   System::Drawing::Point(210,
311);
481   this->labelScanRateHz->Name = L"labelScanRateHz";
482   this->labelScanRateHz->Size = System::Drawing::Size(35, 13);
483   this->labelScanRateHz->TabIndex = 25;
484   this->labelScanRateHz->Text = L"label6";
485   //
486   // buttonRestartScan
487   //
488   this->buttonRestartScan->Location   =   System::Drawing::Point(42,
442);
489   this->buttonRestartScan->Name = L"buttonRestartScan";
490   this->buttonRestartScan->Size = System::Drawing::Size(123, 23);
491   this->buttonRestartScan->TabIndex = 26;
492   this->buttonRestartScan->Text = L"Restart Scan";
493   this->buttonRestartScan->UseVisualStyleBackColor = true;
494   this->buttonRestartScan->Click += gcnew System::EventHandler(this,
&Form1::
buttonRestartScan_Click);
495   //
496   // Form1
497   //
```

```
498   this->AcceptButton = this->button1;
499   this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
500   this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
501   this->AutoSize = true;
502   this->ClientSize = System::Drawing::Size(674, 508);
503   this->Controls->Add(this->buttonRestartScan);
504   this->Controls->Add(this->labelScanRateHz);
505   this->Controls->Add(this->trackBar1);
506   this->Controls->Add(this->progressBar1);
507   this->Controls->Add(this->label1);
508   this->Controls->Add(this->labelScanSize);
509   this->Controls->Add(this->buttonSaveData);
510   this->Controls->Add(this->buttonStopScan);
511   this->Controls->Add(this->buttonStartScan);
512   this->Controls->Add(this->richTextBox2);
513   this->Controls->Add(this->groupBox1);
514   this->Controls->Add(this->statusLabel);
515   this->Controls->Add(this->staticStatusLabel);
516   this->Controls->Add(this->richTextBox1);
517   this->Controls->Add(this->label2);
518   this->Controls->Add(this->textBox1);
519   this->Controls->Add(this->button1);
520   this->Name = L"Form1";
521   this->Text = L"DMSL CMM Laser Scanner Application";
522   this->Load        +=        gcnew        System::EventHandler(this,
&Form1::Form1_Load);
523   this->groupBox1->ResumeLayout(false);
524   this->groupBox1->PerformLayout();
525   (cli::safe_cast<System::ComponentModel::ISupportInitialize^
>(this->trackBar1))->EndInit();
526   this->ResumeLayout(false);
527   this->PerformLayout();
528
529   }
530
531 #pragma endregion
532
533   private:    System::Void    Form1_Load(System::Object^    sender,
System::EventArgs^ e) {
534
535   WSADATA wsaData;
536
537   progressBar1->Maximum = scanCount;
538
539   // Initialize Winsock
540   iResult = WSAStartup(MAKEWORD(2,2), &wsaData);
541   if (iResult != 0) {
542   printf("WSAStartup failed: %d\n", iResult);
543
544   statusLabel->Text = "WSAStartup failed";
545   //return 1;
546   }
547
548   struct addrinfo *result = NULL,
549   *ptr = NULL,
```

```
550   hints;
551
552   ZeroMemory( &hints, sizeof(hints) );
553   hints.ai_family = AF_INET;
554   hints.ai_socktype = SOCK_STREAM;
555   hints.ai_protocol = IPPROTO_TCP;
556
557
558
559   // Resolve the server address and port
560   iResult = getaddrinfo("192.168.0.35", "5010", &hints, &result);
561   if (iResult != 0) {
562   printf("getaddrinfo failed: %d\n", iResult);
563   statusLabel->Text = "getaddrinfo failed:" + iResult;
564   WSACleanup();
565   //    return 1;
566   }
567
568   // Attempt to connect to the first address returned by
569   // the call to getaddrinfo
570   ptr=result;
571
572   // Create a SOCKET for connecting to server
573   ConnectSocket = socket(ptr->ai_family, ptr->ai_socktype,
574   ptr->ai_protocol);
575
576   //check socket is valid
577   if (ConnectSocket == INVALID_SOCKET) {
578   printf("Error at socket(): %ld\n", WSAGetLastError());
579   freeaddrinfo(result);
580   WSACleanup();
581
582   }
583
584   // Connect to server.
585   iResult   =   connect(   ConnectSocket,   ptr->ai_addr,   (int)ptr-
>ai_addrlen);
586   if (iResult == SOCKET_ERROR) {
587   closesocket(ConnectSocket);
588   ConnectSocket = INVALID_SOCKET;
589   }
590
591
592   freeaddrinfo(result);
593
594   if (ConnectSocket == INVALID_SOCKET) {
595   printf("Unable to connect to server!\n");
596   WSACleanup();
597
598   }else{
599   statusLabel->Text = "Connected to CMM Motion Control";
600   }
601
602
603   //begin counter capture card setup
```

```
604   /* Declare UL Revision Level */
605   ULStat = cbDeclareRevision(&RevLevel);
606
607   /* Initiate error handling
608   Parameters:
609   PRINTALL :all warnings and errors encountered will be printed
610   DONTSTOP :program will continue even if error occurs.
611   Note that STOPALL and STOPFATAL are only effective in
612   Windows applications, not Console applications.
613   */
614   cbErrHandling (PRINTALL, DONTSTOP);
615
616   /* set up the display screen */
617
618   /* set the configurable operations of the counter
619   Parameters:
620   BoardNum    :the number used by CB.CFG to describe this board
621   CounterNum  :the counter to be configured (0-5)
622   Quadrature  :Select type of counter input
623   CountingMode      :Slects how counter will operate
624   IndexMode   :Selects what index signal will control
625   InvertIndex :Set to ENABLED id index signal is inverted
626   FlagPins    :Select which signals will drive Flag pins
627   GateEnable  :Set to ENABLED to use external gating signal */
628   CounterNum = 1;
629   //1 = Y, 2 = X, 3=Z
630   Quadrature = X4_QUAD;
631   CountingMode = NORMAL_MODE;
632   DataEncoding = BINARY_ENCODING;
633   IndexMode = INDEX_DISABLED;
634   InvertIndex = DISABLED;
635   FlagPins = CARRY_BORROW;
636   GateEnable = DISABLED;
637
638   //initialize
639
640   ULStat  =  cbC7266Config  (BoardNum,  CounterNum+1  ,  Quadrature,
CountingMode,
641   DataEncoding, IndexMode, InvertIndex, FlagPins,
642   GateEnable);
643   ULStat  =  cbC7266Config  (BoardNum,  CounterNum  ,  Quadrature,
CountingMode,
644   DataEncoding, IndexMode, InvertIndex, FlagPins,
645   GateEnable);
646   ULStat  =  cbC7266Config  (BoardNum,  CounterNum+2  ,  Quadrature,
CountingMode,
647   DataEncoding, IndexMode, InvertIndex, FlagPins,
648   GateEnable);
649
650   /* send a starting value to the counter with cbCLoad()
651   Parameters:
652         BoardNum    :the  number  used  by  CB.CFG  to  describe  this
board
653         RegName     :the  counter  to  be  loading  with  the  starting
value
```

```
654         LoadValue   :the starting value to place in the counter */
655         ////////////////////////////
656         int xLoadValue = 700000;
657         int yLoadValue = 0;
658         int zLoadValue = 0;
659
660         xlastCount = LoadValue;
661         RegName = COUNT1 + CounterNum - 1;
662         printf ("Loading counter #%u with an initial count of %u
using cbCLoad()\n", CounterNum,
      LoadValue);
663         ULStat = cbCLoad32 (BoardNum, RegName, xLoadValue);
664         ULStat = cbCLoad32 (BoardNum, RegName+1, yLoadValue);
665         ULStat = cbCLoad32 (BoardNum, RegName+2, zLoadValue);
666
667         LoadValue = 4000000000;
668         RegName = PRESET1 + CounterNum - 1;
669         printf ("Setting counter #%u maximum count to %u by loading
PRESET register", CounterNum,
      LoadValue);
670         ULStat = cbCLoad32 (BoardNum, RegName, LoadValue);
671         ULStat = cbCLoad32 (BoardNum, RegName+1, LoadValue);
672         ULStat = cbCLoad32 (BoardNum, RegName+2, LoadValue);
673         //end capture card setup
674
675         //Setup Connection to Scanner//
676
677         // initialize Go2 API
678         if ((status = Go2Api_Initialize()) != GO2_OK)
679   {
680   printf("Error: Go2Api_Initialize:%d\n", status);
681   return;
682   }
683   // construct Go2 system object
684   if ((status = Go2System_Construct(&scannerSystem)) != GO2_OK)
685   {
686   printf("Error: Go2System_Construct:%d\n", status);
687   return;
688   }
689   // convert ip address string to Go2IPAddress object
690   if ((status = Go2IPAddress_Parse((const Go2Char *)"192.168.0.45",
&address)) != GO2_OK)
691   {
692   printf("Error: Go2IPAddress_Parse:%d\n", status);
693   return;
694   }
695   // connect to Gocator system at default address as administrator
with empty password
696         if   ((status   =   Go2System_Connect(scannerSystem,address,
GO2_USER_ADMIN, (const Go2Char *) "")) != GO2_OK)
697   {
698   printf("Error: Go2System_Connect:%d\n", status);
699   return;
700   }
701   // establish a Gocator data connection
```

```
702   if   ((status   =   Go2System_ConnectData(scannerSystem,   GO2_NULL,
GO2_NULL)) != GO2_OK)
703   {
704   printf("Error: Go2System_ConnectData:%d\n", status);
705   return;
706   }
707   // stop Gocator system
708   if ((status = Go2System_Stop(scannerSystem)) != GO2_OK)
709   {
710   printf ("Error: Go2System_Stop:%d\n", status);
711   }
712   // start Gocator system
713   if ((status == Go2System_Start(scannerSystem)) != GO2_OK)
714   {
715   printf("Error: Go2System_Start:%d\n", status);
716   return;
717   }
718   Sleep(1000); //added to ensure Scanner is finished System_Start
before calling
Go2System_ReceiveData,
719   //GO2System_ReceiveData fails with the sleep removed.
720   //Allocate Memory
721   memory = (ProfilePoint **) malloc(scanCount * sizeof(ProfilePoint
*));
722   if(memory== NULL)
723   {
724   fprintf(stderr, "out of memory\n");
725   return;
726   }
727   for(i = 0; i < scanCount; i++)
728   {
729   memory[i] = (ProfilePoint *) malloc(1280 * sizeof(ProfilePoint));
730   if(memory[i] == NULL)
731   {
732   fprintf(stderr, "out of memory\n");
733   return;
734   }
735   }
736   //end Scanner setup
737
738   //Gui Initialization
739   labelScanRateHz->Text   =   System::Convert::ToString(   trackBar1-
>Value) + " Hz";
740
741   }
742
743   public:   System::Void   button1_Click(System::Object^   sender,
System::EventArgs^ e) {
744   //printf("Unable to connect to server!\n");
745
746   checkBox1->CheckState                                          =
System::Windows::Forms::CheckState::Unchecked;
747   //add textboxText to
748   richTextBox2->Text = textBox1->Text + "\n" + richTextBox2->Text;
749
```

```
750 #define DEFAULT_BUFLEN 512
751
752   int recvbuflen = DEFAULT_BUFLEN;
753
754   char             *sendbuf             =             (char*)(void*)
Marshal::StringToHGlobalAnsi(textBox1->Text + "\n");
755   char recvbuf[DEFAULT_BUFLEN];
756
757   // Send an initial buffer
758   iResult = send(ConnectSocket, sendbuf, (int) strlen(sendbuf), 0);
759
760   if (iResult == SOCKET_ERROR) {
761   printf("send failed: %d\n", WSAGetLastError());
762   closesocket(ConnectSocket);
763   WSACleanup();
764   //    return 1;
765   }
766
767   // Receive data
768   iResult = recv(ConnectSocket, recvbuf, recvbuflen, 0);
769   String^ clistr = gcnew String(recvbuf);
770   richTextBox1->Text = clistr + richTextBox1->Text + "\n";
771   if (iResult > 0){
772   printf("Bytes received: %d\n", iResult);
773   //label1->Text = L"Bytes received: " + iResult;
774   }
775   else if (iResult == 0)
776   printf("Connection closed\n");
777   else
778   printf("recv failed: %d\n", WSAGetLastError());
779   checkBox1->CheckState                                                =
System::Windows::Forms::CheckState::Checked;
780   }
781
782   private:    System::Void    label3_Click(System::Object^    sender,
System::EventArgs^ e) {
783   }
784
785   private: System::Void GrabScanProfile(){
786
787
788
789   if  (Go2System_ReceiveData(scannerSystem,  RECEIVE_TIMEOUT,  &data)
== GO2_OK)
790   {
791
792   //each result can have multiple data items
793   for (j = 0; j < (signed int)Go2Data_ItemCount(data); ++j)
794   {
795         Go2Data dataItem = Go2Data_ItemAt(data, j); //dataItem seems
to be the single capture frame object
796   if (Go2Object_Type(dataItem) == GO2_TYPE_RAW_PROFILE_DATA)
797   {
798   ProfilePoint16s*                    profileData                    =
(ProfilePoint16s*)Go2RawProfileData_Ranges
```

```
(dataItem);
799   unsigned             int             profilePointCount             =
Go2RawProfileData_Width(dataItem);
800   double XResolution = Go2RawProfileData_XResolution(dataItem);
801   double ZResolution = Go2RawProfileData_ZResolution(dataItem);
802   double XOffset = Go2RawProfileData_XOffset(dataItem);
803   double ZOffset = Go2RawProfileData_ZOffset(dataItem);
804
805         //translate 16-bit range data to engineering units and copy
profiles to memory array
806   for (arrayIndex = 0; arrayIndex < profilePointCount; ++arrayIndex)
807   {
808   if (profileData[arrayIndex].z != -INVALID_RANGE_16BIT)
809   {
810   //This  section  should  be  implemented  using  HTM  of  scanner
orientation
811   memory[scanIndex][arrayIndex].x = XScannerOffset;
812         memory[scanIndex][arrayIndex].y = YScannerOffset - (XOffset
+ XResolution * profileData[arrayIndex].x); //based on the scan stripe
aligned with the Y axis
813         memory[scanIndex][arrayIndex].z = ZScannerOffset + ZOffset +
ZResolution * profileData[arrayIndex].z;
814   }
815   else
816   {
817   memory[scanIndex][arrayIndex].x = INVALID_RANGE_DOUBLE;
818   memory[scanIndex][arrayIndex].y = INVALID_RANGE_DOUBLE;
819   memory[scanIndex][arrayIndex].z = INVALID_RANGE_DOUBLE;
820   }
821   }
822   }
823   }
824
825
826   if ((status = Go2Data_Destroy(data)) != GO2_OK)
827   {
828   printf("Error: Go2Data_Destroy:%d\n", status);
829   return;
830   }
831   } else {
832   printf("Error: Go2System_ReceiveData failed\n");
833   }
834
835
836   scanIndex++;
837
838   //endofSampleScan
839
840   }
841
842   private: System::Void GrabCount(){
843
844   //START COUNTER CAPTURE
845
846   ULStat = cbCIn32 (BoardNum, CounterNum, &xCount);
```

```
847   ULStat = cbCIn32 (BoardNum, CounterNum+1, &yCount);
848   ULStat = cbCIn32 (BoardNum, CounterNum+2, &zCount);
849
850   /* Parameters:
851   BoardNum    :the number used by CB.CFG to describe this board
852   CounterNum :the counter to be setup
853   StatusBits :counter's status returned here */
854   //unused
855   ULStat = cbCStatus (BoardNum, CounterNum, &StatusBits);
856   if (StatusBits & C_UP_DOWN)
857   DirectionStr = "UP ";
858   else
859   DirectionStr = "DOWN";
860
861         //Write X, Y, Z Labels
862         if (xCount != xlastCount){
863         }
864         xlastCount = xCount;
865
866         if (yCount != ylastCount){
867         }
868         ylastCount = yCount;
869
870         if (zCount != zlastCount){
871         }
872         zlastCount = zCount;
873         //End of Counter capture
874
875   }
876
877
878   private:    System::Void    timer1_Tick(System::Object^    sender,
System::EventArgs^ e) {
879   if (timerScan->Enabled == true){
880   checkBox1->CheckState                                          =
System::Windows::Forms::CheckState::Unchecked;
881   }else{
882   GrabCount();
883   }
884   XScannerOffset = (double)xCount / 1000;
885   YScannerOffset = YMax - ((double)(yCount) / 1000) ;
886   ZScannerOffset = ZMax - ((double)(zCount) / 1000) ;
887   xlabel->Text = System::Convert::ToString((double)XScannerOffset);
888   ylabel->Text = System::Convert::ToString(YScannerOffset);
889   zlabel->Text = System::Convert::ToString(ZScannerOffset);
890   }
891   private:   System::Void   checkBox1_CheckedChanged(System::Object^
sender, System::EventArgs^  e) {
892   if                    (checkBox1->CheckState                    ==
System::Windows::Forms::CheckState::Checked){
893   //do something
894   timer1->Enabled = true;
895   }else{
896   //do seomthing else
897   timer1->Enabled = false;
```

```
898   }
899   }
900   private:   System::Void   button2_Click(System::Object^   sender,
System::EventArgs^ e) {
901   // timer1->Enabled = true;
902   LoadValue = 100000;
903   //load value in micrometers
904   int xLoadValue = 700000;
905   //y and z load values declared globally
906   int yLoadValue = 0;
907   int zLoadValue = 0;
908   RegName = COUNT1 + CounterNum - 1;
909         printf ("Loading counter #%u with an initial count of %u
using cbCLoad()\n", CounterNum, LoadValue);
910   ULStat = cbCLoad32 (BoardNum, RegName, xLoadValue);
911   ULStat = cbCLoad32 (BoardNum, RegName+1, yLoadValue);
912   ULStat = cbCLoad32 (BoardNum, RegName+2, zLoadValue);
913   // testCount = 0;
914   // button2->Text = System::Convert::ToString(testCount);
915   }
916
917   private: System::Void textBox1_TextChanged(System::Object^ sender,
System::EventArgs^ e) {
918
919   }
920
921   public: System::Void buttonStartScan_Click(System::Object^ sender,
System::EventArgs^ e) {
922   buttonStartScan->Enabled = false;
923   buttonStopScan->Enabled = true;
924   timerScan->Enabled = true;
925
926   }
927   public: System::Void buttonStopScan_Click(System::Object^ sender,
System::EventArgs^ e) {
928   buttonStopScan->Enabled = false;
929   buttonStartScan->Enabled = true;
930   timerScan->Enabled = false;
931   }
932
933   private:   System::Void   timerScan_Tick(System::Object^   sender,
System::EventArgs^ e) {
934
935   if (scanIndex < scanCount ){
936
937   progressBar1->Value = scanIndex;
938   GrabCount();
939   //update offset
940   XScannerOffset = (double)xCount / 1000;
941   YScannerOffset = YMax - ((double)(yCount) / 1000) ;
942   ZScannerOffset = ZMax - ((double)(zCount) / 1000) ;
943   //
944   Go2System_Trigger(scannerSystem);
945   //
946   GrabScanProfile();
```

```
947   } else{
948   //memory buffer full
949
950   }
951   labelScanSize->Text      =      System::Convert::ToString((unsigned
int)scanIndex);
952   if (trackBar1->Value == 0){
953   buttonStopScan->Enabled = false;
954   buttonStartScan->Enabled = true;
955   timerScan->Enabled = false;
956
957   };
958
959   }
960   private: System::Void buttonSaveData_Click(System::Object^ sender,
System::EventArgs^ e) {
961
962   //save data to file datafile.txt
963   ofstream myfile;
964
965   myfile.open ("ScanData.txt", ios::out | ios::app);
966
967   for (i=0;i < scanIndex ;i++){
968   for (j = 0; j < 1280;j++){
969   if (memory[i][j].x != INVALID_RANGE_DOUBLE)
970         myfile << memory[i][j].x << "," << memory[i][j].y << "," <<
memory[i][j].z << "\n";
971   }
972   }
973   myfile.close();
974
975   }
976
977   private:  System::Void  trackBar1_Scroll(System::Object^  sender,
System::EventArgs^ e) {
978   labelScanRateHz->Text   =   System::Convert::ToString(   trackBar1-
>Value) + " Hz";
979   if (trackBar1->Value != 0){
980   timerScan->Interval = int(1000/trackBar1->Value);
981   }else{
982   timerScan->Interval = int(5);
983   }
984
985   }
986   private:    System::Void    buttonRestartScan_Click(System::Object^
sender, System::EventArgs^  e) {
987   scanIndex = 0;
988   }
989 };
990 }
991
992
```

References

[1] James V. Valentino and Joseph Goldenberg, *Introduction to Computer Numerical Control (CNC)*, 5th ed.: Pearson Education Inc., 2013.

[2] Y. Altintas, I. Yellowley, and J. Tlusty, "The Detection of Tool Breakage in Milling Operations," *Trans. ASME, J. Eng. Ind.*, vol. 110, no. 3, pp. 271-277, August 1988.

[3] R. V. Fleisig and A. D. Spence, "Integrated Digitizing, Path Planning and Five-Axis Machining for the Refurbishment of Compressor and Turbine Blades," *ASME Winter Annual Meeting, Manufacturing Science and Technology*, vol. 6, no. 1, pp. 31-38, 1997.

[4] Robert J. Hocken and Paulo H. Pereira, *Coordinate Measuring Machines and Systems*, 2nd ed. Boca Raton, FL, USA: Taylor & Francis, 2012.

[5] Origin International Inc. (2012) CheckMate, SoftFit Solver: The What, Why & How. [Online]. http://info.originintl.com/Portals/37807/docs/SoftFitSolver_d2.pdf

[6] Yu Pin Lin, "Geometrically Adaptive Milling of Fan Blade Assembly Weld Fillets," Department of Mechanical Engineering, McMaster University, Hamilton, Thesis Paper 7410, 2012.

[7] E. J. Jr Maksym, "Automated Inspection for Flexible Machining Systems," Manufacturing Technology Information Analysis Center,

Chicago, SOAR MTIAC SOAR-88-01, 1988.

[8] G. L Franck, "Flexible inspection systems for automated manufacturing," in *Test, Meas. & Inspect for Quality Control Conference/Exhibition*, 1987.

[9] K. L. Blaedel, "Error Reduction," University of CA, Livermore, CA, Report of the Machine Tool Task Force, Volume 5: Machine Tool Accuracy UCRL-52960-S, 1980.

[10] ISO, "Geometrical Product Specifications (GPS) -- Acceptance and reverification tests for coordinate measuring machines (CMM) -- Part 2: CMMs used for measuring size," ISO 10360-2:2001 , 2001.

[11] ISO, "Test code for machine tools -- Part 2: Determination of accuracy and repeatability of positioning numerically controlled axes," ISO 230-2:2006, 2006.

[12] ISO, "Test code for machine tools -- Part 10: Determination of the measuring performance of probing systems of numerically controlled machine tools," ISO 230-10:2011, 2011.

[13] J. Bryan, "Design and Construction of an 84 inch diameter diamon turning machine," *Precision Engineering*, vol. 1, no. 13, pp. 13-17, 1979.

[14] W. T. Estler and E. B. Magrab, "Validation metrology of the large optics diamond turning machine," National Bureau of Standards Interagency

Report NBSIR-85-3182R, 1985.

[15] S. R. 1986 Patterson, "Development of Precision Turning Capabilities at Lawrence Livermore National Laboratory," in *3rd Biennial International Machine Tool Technical Conference*, Chicago, IL, 1986, pp. 147-59.

[16] K. Kim, K. Eman, and S. M. Wu, "In-process control of cylindricity in boring operations," *ASME Trans. Journal of Engineering for Industry*, vol. 109, no. 4, pp. 385-391, 1987.

[17] M. A. Donmez, "A real-time control system for CNC machine tool based on deterministic metrology.," in *Statistical Process Control in Automated Manufacturing*, J. B. Keats and N. F. Hubels, Eds. New York: Marcel Dekker, Inc., 1989.

[18] J. Bryan, "The deterministic approach in metrology and manufacturing," in *Proceedings of the 1993 International Forum on Dimensional Tolerancing and Metrology*, New York, 1993, pp. 85-95.

[19] G et al. Belforte, "Coordinate measuring machines and machine tools self-calibration and error correction," *Annals of CIRP*, vol. 36, no. 1, pp. 359-364, 1987.

[20] R. Hocken, J. Raja, and U. Babu, "Sampling Issues in Coordinate Metrology," *Manufacturing Review*, vol. 6, p. 282, 1993.

[21] "International vocabulary of metrology – Basic and general concepts and associated terms (VIM), 3rd Edition," Joint Committee for Guides in Metrology, JCGM 200:2012, 2012.

[22] J.A. Bosch, *Coordinate Measuring Machines and Systems*, G., Dieter, G. Boothroyd, Ed.: Marcel Dekker Inc., 1995.

[23] Renishaw Plc. (2002, December) SP600 Scanning Probe System. [Online]. http://www.renishaw.com/en/sp600--6687

[24] Renishaw Plc. (2012, August) Touch-trigger Probes. [Online]. http://www.renishaw.com/en/touch-trigger-probes--6652

[25] Fanuc Ltd., "Fanuc Series 15i-MA Operator's Manual, B-63324EN/01," Manual 1998.

[26] R. Szelisiki,.: Springer, 2010, ch. 7, pp. 345-347.

[27] Wikipedia. (2008, May) Structured Light 3D Scanner. [Online]. http://en.wikipedia.org/wiki/Structured-light_3D_scanner

[28] Altschuler et al., "Robot Vision by Encoded Light Beams," in *Three-Dimensional Machine Vision*.: Kluwer Academic Publishers, 1987, ch. 1, p. 99.

[29] ShapeGrabber. (2012, November) Shapegrabber company Web site. [Online]. http://blog.shapegrabber.com/2012/05/optical-3d-inspection-of-transparent-or-reflective-parts.html

[30] LMI 3D. (2012, August) Line Triangulation. [Online]. http://www.lmi3d.com/education/line-triangulation

[31] G. T. Anthony and M. G. Cox, "Reliable Algorithms for Roundness Assessment to BS3730," in *Software for Coordinate Measuring Machines*, M. G. Cox and G. N. Peggs, Eds. Teddington, UK: National Physical Laboratory, 1986, pp. 30-37.

[32] K.S. Arun, T.S. Huang, and S.D. Blostein, ""Least-Squares Fitting of Two 3-D Point Sets," *IEEE Trans. PAMI*, vol. 9, no. 5, pp. 698-700, 1987.

[33] P.J. Besl and H.D. McKay, "A Method for Registration of 3-D Shapes," *IEEE TRANS. PAMI*, vol. 14, no. 2, pp. 239-256, 1992.

[34] Geomagic, Inc. (2013, February) Geomagic Studio. [Online]. http://www.geomagic.com/en/products/studio/overview/

[35] InnovMetric Software Inc. (2013, March) PolyWorks Solutions - Point Cloud Engineering. [Online]. http://www.innovmetric.com/polyworks/3D-scanners/so_pointcloud3.aspx?lang=en

[36] Origin International Inc. (2012, Nov.) Origin International, Inc. [Online]. http://info.originintl.com/Portals/37807/docs/ISO.pdf

[37] Radu Bogdan Rusu and Steve Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation*

(ICRA), Shanghai, China, 2011.

[38] Memex Automation Inc. a unit of Astrix Networks Inc. (2012, August) Ax9150 Universal Machine Interface (UMI). [Online]. http://memex.ca/index.php?option=com_content&view=article&id=46&Itemid=200145

[39] P. Smid, *Fanuc CNC custom macros: Programming resources for Fanuc Custom Macro B users*, Illustrated ed.: Industrial Press Inc., 2004.

[40] Renishaw PLC. (2011) (TE412) One-Touch vs two-touch probing strategies. [Online]. http://resources.renishaw.com/download.aspx?data=32423(=&showForm=true

[41] Vero Software Limited. (2013) EdgeCam. [Online]. www.edgecam.com

[42] Roland DGA Corporation. (2012, September) LPX-60DS/600DS/1200DS 3D Laser Scanners Specifications. [Online]. http://www.rolanddga.com/products/scanners/lpx600/#specifications

[43] C. Y. Wu, "Arbitrary Surface Flank Milling and Flank SAM in the Design and Manufactoruing of Jet Engine and Compressor Coils," in *ASME Turbo Expo 2012*, Copenhagen, 2012, p. 2.

[44] LMI Technologies. (2012, May) LMI Technologies. [Online]. www.lmi3D.com

[45] Renishaw Plc. (2012, August) Inspection Plus - software for machining centres. [Online]. http://www.renishaw.com/en/inspection-plus-software-for-machining-centres--6094

[46] Rolls-Royce. (2003, April) Rolls-Royce Ships First Blisk for Joint Strike Fighter SDD Programme. [Online]. http://www.f-16.net/f-16_forum_viewtopic-t-9880-start-30.html

[47] D. Alan Sawula and Allan D. Spence, "Integrated CNC Measurement, Analysis and Tool Path Adjustment," in *Virtual Machining Process Technology*, vol. 1, Montreal, 2012.

[48] Allan D. Spence, D. Alan Sawula, James R. Stone, and Yu Pin Lin, "In-situ Measurement and Distributed Computing for Adjustable CNC Machining," in *Computer-Aided Design and Applications*, Bergamo, Italy, 2013.